README FILE OF YNOGK

This is the readme file for code YNOGK (YunNan Observatory Geodesics Kerr). YNOGK is a public code that calculates four Boyer-Lindquist coordinates (r, θ, ϕ, t) and affine parameter σ in Kerr spacetime. For the reference one can see paper of Xiao-lin, Yang & Jian-cheng, Wang (2012). You are morally obligated to cite this paper in your scientific literature if you used the code in your research.

This package contains the source file of the code: ynogk.f90, several examples which demonstrate the reader how to use this source code; and idl programs, which can plot figures from data calculated by the code; and the software g95, the reader is recommended to use g95 to compile ynogk.f90.
****

Source file ynogk.f90 is composed by seven modules, they are:

1. module **constants**—which defines many constants often used in the program.

2. module **rootsfinding**—Which contains subroutines root3 and root4, they can solve cubic and quartic equations and return the roots of them.

3. module **ellfunction**—Which includes supporting subroutines and functions to compute Weierstrass' and Jacobi's elliptical functions and integrals, especially the subroutines for Carlson's integrals.

4. module **BLcoordinates**—Which contains supporting subroutines and functions to compute four Boyer-Lindquist coordinates (r, θ, ϕ, t) and affine parameter σ.

5. module **sphericalmotion**—Which contains the routines to compute the spherical orbit of a photon in a Kerr spacetime.

6. module **pem-finding**—Which includes functions to search for p_{em}, which is the root of equation $f(p) = 0$. And $f(p)$ is a single variable function, which describes the surface of the emission region. To see the definition of $f(p)$ one can cf. our paper.

7. module **obsemitter**—Which contains functions and subroutines to determine the geodesic connecting the observer and emitter.

In many applications one may encounter an special quantity, i.e., the infinity, in our code we use a special constant "**infinity**" to represent ∞, and "**infinity**$=10^{40}$". If a result is equal or bigger than "10^{40}", the code will treat it as ∞, else it will be treated as a finite number. One can reset this value to satisfy their own needs.

****

We give five examples to demonstrate the reader how to use the source code to literature problems.

The first one is to draw the geodesic trajectories of a group of photons emitted isotropically from a particle orbits around a black hole (a=0.95) at a Keplerian orbit. The source files are in directory ./rays. To get start one can compile the file ray.f90 by following command:

```
[@localhost rays]$ g95 ray.f90 -o ray
[@localhost rays]$ time ./ray <data.in
```

where other parameters are given in file data.in.

then one can use the IDL's commands to draw the figure:

```
IDL> .r raylines.pro
IDL> raylines
```

The second one is to draw the images of the thin disk and the source file are given in directory ./thin disk. Using following command one can compile and run the file thindisk.f90:

```
[@localhost thin disk]$ g95 thindisk.f90 -o thindisk
[@localhost thin disk]$ time ./thindisk <data.in
```

where other parameters are given in file data.in.

To draw the image one only need to run following command:

```
IDL> .r thind_axis.pro
IDL> thind_axis
```

The other three examples are to draw the images of warped disks, a shadow of black hole and a circular orbits of a photon. The source files are given in directory ./warped disk, ./shadows and /circularorbits respectively. The commands to compile and run the source files are similar with above examples.

****

In YNOGK the four B-L coordinates (r, θ, ϕ, t) and affine parameter σ has been expressed as functions of a parameter p, which is always nonnegative. In the source code function r

and $\mu = \cos\theta$ have following forms:

$$\textbf{radius}(\mathrm{p}|\mathrm{f1234(1), lambda, q, aspin, robs, scal})$$
$$\textbf{mucos}(\mathrm{p}|\mathrm{f1234(3), f1234(2), lambda, q, sinobs, muobs, aspin, scal})$$

Where p is the independent variable, other parameters should be supplied before call this two functions. Obviously

"**aspin**" is the black hole spin: a, which satisfies $-1 < a < 1$.

"**sinobs**"=$\sin\theta_{obs}$, "**muobs**"=$\cos\theta_{obs}$. θ_{obs} is the inclination angle of the observer or the initial θ angle of the photon. We recommend the reader to use following code section to compute sinobs and muobs

```
If(cobs.ne.90.D0)then
    cobs = cobs * dtor
    muobs = cos(cobs)
    sinobs = sin(cobs)
else
    muobs = 0.D0
    sinobs = 1.D0
endif
```

"**robs**" is the radial coordinate of the observer or the initial position of the photon.

"**scal**" is a parameter to control the size of the image. Which is simply set to 1.

"**lambda**" and "**q**" are motion constants: λ and q. "**f1234**" is an array, the definitions of which are equations (106)-(109) in our paper, and f1234(1)="f_1", f1234(2)="f_2", f1234(3)="f_3", f1234(4)="f_4". "**lambda**", "**a**" and array "**f1234**" can be computed by two subroutines "**lambdaq**" and "**initialdirection**" in our code, which have following form:

$$\textbf{lambdaq}(\mathrm{alpha, beta, robs, sinobs, muobs, aspin, scal, velocity, f1234, lambda, q}).$$

Subroutine "**lambdaq**" can compute motion constants "**lambda**" and "**q**" and array "**f1234**(1:4)" from impact parameter α, β, where "alpha" and "beta" are impact parameters one should provide. Another quantity one needs to provide is the physical velocity of the observer with respect to LNRF, i.e. an array "velocity(1:3)", and velocity(1)=v_r, velocity(2)=v_θ, velocity(3)=v_ϕ. If the observer is at infinity and keep stationary, then one can set velocity(1)=0, velocity(2)=0, velocity(3)=0. To Compute constants of motion from impact parameter mainly used in imaging, and spectra or line profile calculation.

$$\textbf{initialdirection}(\mathrm{pr, ptheta, pphi, sinobs, muobs, aspin,} \\ \mathrm{robs, scal, velocity, lambda, q, f1234}).$$

Subroutine "initialdirection" can compute motion constants "lambda", "q" and array "f1234(1:4)" from initial four momentum of photon: p'_r, p'_θ, p'_ϕ, which are tested by the emitter in its local rest frame, and pr=r'_r, ptheta=p'_θ, pphi=p'_ϕ. And the motion state of the emitter is described by its physical velocity with respect to LNRF. And the velocities are another quantities one should specify. Which is the an array "velocity(1:3)", and velocity(1)=υ_r, velocity(2)=υ_θ, velocity(3)=υ_ϕ.

With parameter aspin, sinobs, muobs, robs, scal, lambda, q and array f1234(1:4), one can call function

$$\textbf{radius}(\mathrm{p|f1234(1), lambda, q, aspin, robs, scal})$$
$$\textbf{mucos}(\mathrm{p|f1234(3), f1234(2), lambda, q, sinobs, muobs, aspin, scal})$$

to compute r and μ coordinates of a geodesic for a p was given. It is a situations in which we assuming one only need to compute r and μ coordinate. For example applications are axisymmetric. To do so, one can save running time of the code.

Meanwhile we supply another subroutine "**ynogk**" to compute the four B-L coordinates (r, θ, ϕ, t) and affine parameter σ. "ynogk" has following form:

$$\textbf{ynogk}(\mathrm{p, f1234, lambda, q, sinobs, muobs, aspin, robs, scal, radi, mu, phi, time, sigma}).$$

where "p, **f1234, lambda, q, sinobs ,muobs, aspin, robs, scal**" are parameters one should specify before call **ynogk**, and "radi, mu, phi, time, sigma" are five return variables, which are the four B-L coordinates and affine parameter.
****

As we have discussed in our paper, in the application of imaging, if the observer has velocity, then the images on the screen of the observer will has a displacement, which is proportional to the velocity. And the displacement can be described by a vector on the screen (α_c, β_c). α_c and β_c satisfy following equations

$$f_3(\alpha_c, \beta_c) = 0,$$
$$f_2(\alpha_c, \beta_c) = 0.$$

Obviousely, $f_3(\alpha_c, \beta_c)$ describes the projection of spin axis of black hole onto the screen of observer, and $f_2(\alpha_c, \beta_c)$ describes the projection of a line, which is in the equatorial plane of the black hole and perpendicular to a constant ϕ plane, in which the observer locating. we supply a subroutine "center-of-image" to solve above equations

$$\textbf{centerofimage}(\text{robs}, \text{sinobs}, \text{muobs}, \text{aspin}, \text{scal}, \text{velocity}, \text{alphac}, \text{betac})$$

which will return "alphac" and "betac", that is α_c and β_c. In real applications the impact parameter should be confined in region $(-\Delta L + \alpha_c, \Delta L + \alpha_c)$ and $(-\Delta L + \beta_c, \Delta L + \beta_c)$, where ΔL is the half length of the image. Otherwise one may not find the images on the screen outside this region.
****

In module pemfinding, we provide a function "pemfind" as follows to get the root of equation $f(p) = 0$.

$$\textbf{pemfind}(\text{f1234}, \text{lambda}, \text{q}, \text{sinobs}, \text{muobs}, \text{aspin}, \text{robs}, \text{scal}, \text{rin}, \text{rout}, \text{muup},$$
$$\text{mudown}, \text{phi1}, \text{phi2}, \text{caserange}, \text{Fp}, \text{paras}, \text{bisection}),$$

where Fp is the name of function $f(p)$, which should be provided by the user. And Fp has a following header

$$\textbf{Fp}(\text{p}, \text{f1234}, \text{lambda}, \text{q}, \text{sinobs}, \text{muobs}, \text{aspin}, \text{robs}, \text{scal}, \text{t1}, \text{t2}, \text{paras})$$

where t1 and t2 are number of the photon meeting turning points. paras is an array: paras(1:10), which transfers parameters may be used in Fp.

rin and rout are the inner and outer radius of the surface described by $f(p)$. Similarly muup and mudown, phi1 and phi2 are the minimum and maximum of μ and ϕ coordinates of the surface. muup and mudown, phi1 and phi2 are optional parameters, user set caserange=1 to tell the function that all of them have been provided, and caserange=2 if muup and mudown are provided but phi1 and phi2 are not provided, caserange=3, if muup and mudown are not provided but phi1 and phi2 are provided, and caserange=4 if all of them are not provided. Usually caserange was set to be 4.

bisection is a logical variable, if bisection=.TRUE., then the code will use bisection method to search the root of equation $f(p) = 0$, else the Newton-Raphson method will be used.

In the subroutine **pemfind** there is an important parameter: NN, which is the number of sections of the interval (p_1, p_2) or (p_3, p_4) has been divided when searching the roots. One needs to set a proper value for NN, since if NN is too small, the roots exit on the interval (p_1, p_2) or (p_3, p_4) maybe omitted, if NN is too big, the time consumed by the code will be large.

****
In module obsemitter we provide a subroutine "alpha-beta-p" to determine a geodesic connecting the observer and emitter, with the coordinates of them were specified, i.e. to solve the following equations for $\alpha_{em}, \beta_{em}, p_{em}$ by Newton-Raphson method

$$\begin{aligned} \mathrm{r}(\alpha_{\mathrm{em}}, \beta_{\mathrm{em}}, \mathrm{p}_{\mathrm{em}}) &= \mathrm{r}_{\mathrm{em}}, \\ \mu(\alpha_{\mathrm{em}}, \beta_{\mathrm{em}}, \mathrm{p}_{\mathrm{em}}) &= \mu_{\mathrm{em}}, \\ \phi(\alpha_{\mathrm{em}}, \beta_{\mathrm{em}}, \mathrm{p}_{\mathrm{em}}) &= \phi_{\mathrm{em}}. \end{aligned}$$

where $r_{em}, \mu_{em}, \phi_{em}$ is the coordinates of the emitter. Subroutine "alpha-beta-p" has following parameters

$$\textbf{alphabetap}(\mathrm{a0, B0, alen, Blen, sinobs, muobs, aspin, robs, scal, obsV, rmuphyem,} \\ \mathrm{abp, func1, func2, func3}).$$

This subroutine will return the impact parameters of the geodesic α_{em}, β_{em} and the parameter p$_{em}$ corresponds to the position of the emitter through an array abp(1:3), and abp(1)=α_{em}, abp(2)=β_{em}, abp(3)=p_{em}. The coordinates of observer are θ_{obs}, r_{obs}, the coordinates of the emitter should provide by the user through an array rmuphyem(1:3), and rmuphyem(1)=r_{em}, rmuphyem(2)=$\cos\theta_{em}$, rmuphyem(3)= ϕ_{em}. The subroutine will search the α_{em} and β_{em} in a retangular region on the screen of the observer, parameter alen and Blen are the size of the retangular, and a0, B0 is the coordinates of one corner the retangular. func1, func2, func3 are names of function radius(p), mucos(p) and phi(p).

For the spherical motion, Shakura in 1987 has discussed the computation of integral constants from the r_{sm}, θ_{min}, a, which are the radius, the minimum of θ coordinate of the motion respectively, and a is the black hole spin. He given in his paper that

$$\begin{aligned} \frac{L}{m} &= \frac{F_L(r_{sm}, \theta_{min}, a)}{D(r_{sm}, \theta_{min}, a)}, \\ \frac{E}{m} &= \frac{F_E(r_{sm}, \theta_{min}, a)}{D(r_{sm}, \theta_{min}, a)}, \\ \frac{Q}{m^2} &= \frac{F_Q(r_{sm}, \theta_{min}, a)}{D(r_{sm}, \theta_{min}, a)}. \end{aligned}$$

For a photon whose rest mass m is zero, which gives $D(r_{sm}, \theta_{min}, a) = 0$. So when r_{ms}, a are given, θ_{min} is totally determined, i.e. $\theta_{min} = \theta_{min}(r_{ms}, a)$. And the constants of the spherical motion are given by

$$\lambda = \frac{L}{E} = \frac{F_L(r_{sm}, a)}{F_E(r_{sm}, a)},$$
$$q = \frac{Q}{E^2} = \frac{F_Q(r_{sm}, a)}{F_E(r_{sm}, a)^2}.$$

which can be computed by the routine named **lambdaqsphericalm**.
****

If you find any bugs or have any questions about ynogk please sent a email to me. My email address is: yangxl@ynao.ac.cn.