RELAGN: Documentation

Scott Hagen
Email: scott.hagen@durham.ac.uk

Contents

1 RELAGN

We will start by describing the main model RELAGN. Throughout we assume you have either downloaded or cloned the GitHub repository, and we will work on the assumption that you have not changed the directory structure within the repository since downloading it.

This documentation is only meant as a guide on how to use the code. For a description of the model, please see the main paper (Hagen & Done, in prep). If this code us useful in your work, please cite: Input bibtex for paper here!!

1.1 Input Parameters

Here we give an overview of the parameters that define the model. We include a brief description, the units, and the defualt values (i.e what the code will use if you do not pass this parameter). For some parameters we also include limits. These are **not based off any physical argument** - but actual limits that will break the code if exceeded (for a variety of reasons). For an idea of sensible **physical** limits, see Hagen & Done, insert here for a description of the physics that go into the model.

Par 1: M Mass of central black hole
- *Units*: $M_\odot$
- *Default*: 10^8

Par 2: D Distance from the observer to the black hole
- *Units*: Mpc
- *Defualt*: 100
- *Limits*: $D > 0$ - Must be greater than 0 distance...

Par 3: $\log \dot{m}$ Mass-accretion rate - Scaled by the Eddington mass accretion rate, such that $\dot{m} = \dot{M}/\dot{M}_{\rm Edd}$, where $\dot{M}$ is the physical mass accretion rate of the system (i.e unit mass per unit time) and $\dot{M}_{\rm Edd}$ is the Eddigton mass accretion rate. This is related to the Eddington luminosity through $L_{\rm Edd} = \eta \dot{M}_{\rm Edd} c^2$, where η is a black hole spin dependent efficiency factor, and c is the speed of light.
- *Units*: Dimensionless
- *Default*: -1

Par 4: a Black hole spin parameter. 0 Implies non-spinning, while 1 is maximally spinning with prograde rotation (i.e in the same direction as the accretion disc). Note that the code enforces an upper limit of 0.998, which is the theoretical maximum assuming the presence of a disc insert Thorne 1974 citation!
- *Units*: Dimensionless
- *Default*: 0
- *Limits* $0 \leq a \leq 0.998$ (Retrograde rotation not currently supported by the GR transfer functions we use)

Par 5: $\cos(i)$ Cosine of the inclination of the observer with respect to the disc. This is measured from the z-axis, with the disc in the x-y plane (i.e $\cos(i) = 1$ would imply an observer located on the z-axis looking straight down onto the disc, while $\cos(i) = 0$ will imply an edge on view of the disc).
- *Units*: Dimensionless
- *Default*: 0.5
- *Limits*: $0.09 \leq \cos(i) \leq 1$ (Exactly edge on will give you a disc that is not visible...)

Par 6: $kT_{e,h}$ Electron temperature for the hot Comptonisation region (i.e the X-ray corona). This sets the high-energy roll-over of hot Comptonisation component in the spectrum.

- *Units*: keV
- *Default*: 100
- *Limits*: $0 < kT_{e,h}$ (Apart from being wildly unrealistic, 0 electron temperature will lead to segmentation faults)

Par 7: $kT_{e,w}$ Electron temperature for the warm Componisation region
- *Units*: keV
- *Default*: 0.2
- *Limits*: $0 < kT_{e,w}$ (Same reasoning as above!)

Par 8: Γ_h Spectral index for the hot Comptonisation component
- *Units*: Dimensionless
- *Default*: 1.7
- *Limits*: $1.1 \leq \Gamma_h$ (NTHCOMP will break for unrealistically steep spectra)

Par 9: Γ_w Spectral index for the warm Comptonisation component
- *Units*: Dimensionless
- *Default*: 2.7
- *Limits*: $1.1 \leq \Gamma_w$ (Same reasoning as above)

Par 10: r_h Outer radius of the hot Comptonisation region (X-ray corona). If this is negative, then the code will use the innermost stable circular orbit, $r_{\rm isco}$
- *Units*: Dimensionless gravitational units, $r = R/R_G$ where $R_G = GM/c^2$
- *Default*: 10

Par 11: r_w Outer radius of the warm Comptonisation region. If this is negative, then the code will use $r_{\rm isco}$.
- *Units*: R_G
- *Default*: 10

Par 12: $\log r_{\rm out}$: Outer disc radius. If this is negative, then the code will use the self-gravity radius from Insert Laor and Netzer ref here.
- *Units*: R_G
- *Default*: -1 (i.e self-gravity)

Par 13: $f_{\rm col}$ Colour-temperature correction. Note, that this is only applied to the standard disc region. If negative, then the code will use the relation given in insert Done et al 2012 reference here!!. Otherwise it is treated as a constant correction across the entire standard disc region, such that the black-body emission from each annulus is given by $B_\nu(f_{\rm col}T(r))/f_{\rm col}^4$, where $T(r)$ is the temperature at that the annulus and B_ν denotes the black-body emission.
- *Units*: Dimensionless
- *Default*: 1

Par 14: $h_{\rm max}$ Maximal scale-height of the hot X-ray corona. This is a tuning parameter, and only affects the seed photons from the disc intercepted by the corona. If $h_{\rm max} > r_h$, then

the code will automatically switch to r_h as the maximal scale-height.
- *Units*: R_G
- *Default*: 10

Par 15: z Redshift of the source (i.e the black hole) as seen by the observer. As all calculations are initially done in the frame of the black hole, this correction is only applied when transforming to an observed spectrum.
- *Units*: Dimensionless
- *Default*: 0

1.2 Running through XSPEC

If you wish to fit the model to observational data the easiest way is through XSPEC, as this will take into account telescope effective areas and responses. To this extent, we have written a bespoke XSPEC version of the model in FORTRAN. Before getting started though, there are a couple of steps to required by you in order to compile the model.

1.2.1 Installation and Compilation

As the code is written for XSPEC it should also be compiled within XSPEC. To make this simple we have included a shell script, COMPILE_TO_XSPEC.SH, which executes the required commands for compilation. This is found within the main RELAGN directory and is excecuted by typing:

```
> sh compile_to_xspec.sh
```

while within the RELAGN main directory. Note that this will compile both RELAGN and RELQSO. What this does is execute the following commands (which you can type manually if you wish, instead of using the shell script):

```
> xspec
> initpackage relagn lmod_relagn.dat /Path/To/RELAGN/src/fortran_version/relagn_dir
```

where /Path/To is a place-holder for the directory path to RELAGN. The code should now be compiled (you should check the terminal for any big errors!). If the compilation was successful, then you do not need to repeat this step - ever! (unless you happen to delete or move the source code).

The next step is to load the compiled code as a local model. This is done from within XSPEC. So within your terminal, type:

```
> xspec
> lmod relagn /Path/To/RELAGN/src/fortran_version/relagn_dir
```

The model is now loaded, and you are good to go! Enjoy! (Note, that you will need to load it into XSPEC **every** time, unless you append to your XSPEC.RC file. More on that below. If you want more information regarding compiling and loading local models in XSPEC, taken a look at the XSPEC documentation (https://heasarc.gsfc.nasa.gov/xanadu/xspec/manual/XSappendixLocal.html)

Automatically loading RELAGN upon starting XSPEC :

If, like me, you do not want to have to type LMOD ETC... every time you wish to use the model in a new XSPEC session, you can modify your XSPEC.RC file. This file contains any commands you wish XSPEC to execute upon startup, and is located in the ~/.XSPEC directory (I'm assuming you compiled HEASOFT using the source code, and following the instructions, and so this directory **should** exist within you home directory.).

Now, cd into the ~/.XSPEC directory, and open the xspec.rc file. If this file doesn't exist, create one. Within the file, add the line:

```
lmod relagn /Path/To/RELAGN/src/fortran_version/relagn_dir
```

XSPEC will now automatically execute that command upon start-up. Don't want to have to type all that out yourself? No problemo, we've also included a shell script that will modify your xspec.rc file accordingly - so no typos! From within the RELAGN directory, simply type:

```
> sh init_autoLoad_xspec.sh
```

This will **append** the lmod line into your xspec.rc file (Note it will append for **both** RELAGN and RELQSO). Only run this if you want XSPEC to automatically load both models **every** time you start a new session!!! For more info on modifying XSPEC, take a look at their documentation (https://heasarc.gsfc.nasa.gov/xanadu/xspec/manual/node33.html)

1.2.2 Fitting to data - Worked example

1.3 Running through PYTHON

Occasionally you might not wish to run the model through XSPEC. For example, you could be using the model as a part of one of your own codes/models, or you simply don't enjoy using XSPEC for your data analysis. To this extent using the PYTHON version makes more sense (incidentally this was the original version of the model. The FROTRAN version only came into existence when I wanted an easier way of making it work with XSPEC...)

In PYTHON the RELAGN model exists as a class that you can initiate (by passing your desired input parameters), and then you choose what parts of the model you want to calculate/extract (e.g you can choose to skip the GR ray tracing calculations, or only calculate specific components of the SED, etc.). Below we start by detailing the RELAGN class attributes and methods, after which we give a few examples on creating and extracting SEDs.

1.3.1 RELAGN class description

```
Help on relagn in module relagn object:

class relagn(builtins.object)
 |   relagn(M=100000.0, dist=1, log_mdot=-1, a=0, cos_inc=0.5, kTe_hot=100, kTe_warm=0.2, g
 |
 |   relagn - relativistic SED model for AGN
 |   Assumes a radially stratified flow consisting of:
```

```
|       - An outer standard accretion disc
|       - A warm Comptonisation region (where the disc is having a bad day)
|       - A hot Comptonisation region (i.e inner X-ray Corona)
|
|   For more details on the model - see Hagen & Done (in prep)
|
|
|
|   Attributes
|   ----------
|   risco : float
|       Innremost stable circular orbit - units : Dimensionless (Rg)
|
|   r_sg : float
|       Self-Gravity radius of the disc, following Laor & Netzer 1989
|       Units : Dimensionless (Rg)
|
|   eta : float
|       Accretion efficiency. Used to convert from mass accretion rate to
|       luminosity. i.e L = eta * Mdot * c^2
|       Units : dimensionless
|
|   Egrid : array
|       Energy grid used for calculations (in frame of black hole) - units : keV
|
|   nu_grid : array
|       Frequency grid corresponding to Egrid - units : Hz
|
|   wave_grid : array
|       Wavelength grid corresponding to Egrid - units : Angstrom
|
|   E_obs : array
|       Energy grid converted to observer frame (i.e redshift corrected) - units : keV
|
|   nu_obs : array
|       Frequency grid converted to observer frame - units : Hz
|
|   wave_obs : array
|       Wavelength grid converted to observer frame - units : Angstrom
|
|   ###########################################################################
|   There are other attributes, however it is recomended that you
|   extract these using the built in methods in order to deal with unit choices
|   correctly. The remaining ones after this are only necessary for the internal
|   calculations, so no need to worry about them!
|   ###########################################################################
|
|
|   Important Methods
```

```
|   ______________________
|   get_totSED(rel=True)
|       Extracts the total SED (disc + warm + hot components) in whatever
|       units are set
|
|   get_DiscComponent(rel=True)
|       Extracts disc component from SED (after checking if it exists in the
|       current model geometry) in whatever units are set
|
|   get_WarmComponent(rel=True)
|       Extracts warm Compton component from SED (after checking if it exists
|       in the current model geometry) in whatever units are set
|
|   get_HotComponent(rel=True)
|       Extracts hot Compton component from SED (after checking if it exists
|       in the current model geometry) in whatever units are set
|
|   set_units(new_unit='cgs')
|       Sets the system units to use when extracting spectra / system
|       properties
|
|   set_flux()
|       Sets a flag s.t all spectra are given in terms of flux rather than
|       luminosity
|
|   set_lum()
|       Sets a flag s.t all spectra are given in luminosity (this is the default)
|
|   get_Ledd()
|       Gives Eddington luminosity in whatever units are set
|       (Note: in frame of black hole!!)
|
|   get_Rg()
|       Gives scale of gravitaional radius (Rg = GM/c^2) in whatever units
|       are set
|
|   get_Mdot()
|       Gives PHYSICAL mass accretion rate, in either g/s or kg/s
|       (depending on what units are set)
|
|
|   ##############################################################################
|
|   Methods defined here:
|
|   Lseed_hotCorona(self)
|       Calculates luminsoty of seed photons emitted at radius r, intercepted
|       by corona
|
```

```
|       Returns
|       -------
|       Lseed_tot : float
|           Total seed photon luminosity seen by corona - units : W
|
|   __init__(self, M=100000.0, dist=1, log_mdot=-1, a=0, cos_inc=0.5, kTe_hot=100, kTe_warm
|       Parameters
|       ----------
|       M : float
|           Black hole mass - units : Msol
|       dist : float
|           Co-Moving Distance - units : Mpc
|       log_mdot : float
|           log mass accretion rate - units : Eddington
|       a : float
|           Dimensionless Black Hole spin - units : Dimensionless
|       cos_inc : float
|           cos inclination angle
|       kTe_hot : float
|           Electron temp for hot Compton region - units : keV
|       kTe_warm : float
|           Electron temp for warm Compton region - units : keV
|       gamma_hot : float
|           Spectral index for hot Compton region
|       gamma_warm : float
|           Spectral index for warm Compton region
|       r_hot : float
|           Outer radius of hot Compton region - units : Rg
|       r_warm : float
|           Outer radius of warm Compton region - units : Rg
|       log_rout : float
|           log of outer disc radius - units : Rg
|       fcol : float
|           Colour temperature correction as described in Done et al. (2012)
|           If -ve then follows equation 1 and 2 in Done et al. (2012).
|           If +ve then assumes this to be constant correction over entire disc region
|       h_max : float
|           Scale height of hot Compton region - units : Rg
|       z : float
|           Redshift
|
|   calc_Tnt(self, r)
|       Calculates Novikov-Thorne disc temperature^4 at radius r.
|
|       Parameters
|       ----------
|       r : float OR array
|           Disc radius (as measured from black hole)
|           Units : Dimensionless (Rg)
```

```
|
|       Returns
|       -------
|       T4 : float OR array
|           Novikov-Thorne disc temperature at r (to the power 4)
|           Units : K^4 (Kelvin to power 4)
|
|   calc_fcol(self, Tm)
|       Calculates colour temperature correction following Eqn. (1) and
|       Eqn. (2) in Done et al. (2012)
|
|       Parameters
|       ----------
|       Tm : float
|           Max temperature at annulus (ie T(r)) - units : K.
|
|       Returns
|       -------
|       fcol_d : float
|           colour temperature correction at temperature T
|
|   change_rBins(self, new_drdex)
|       JUST FOR TESTING PURPOSES!!!! Allows changing of radial bin-width
|       makes testing easier...
|
|       Parameters
|       ----------
|       new_drdex : float
|           New number of steps per decade.
|
|   disc_annuli(self, r, dr)
|       Calculates disc spectrum for annulus at position r with width dr. Note
|       that r is taken to be the center of the bin!
|
|       Parameters
|       ----------
|       r : float
|           Inner radius of annulus - units : Rg.
|       dr : float
|           Width of annulus - units : Rg.
|
|       Returns
|       -------
|       Lnu_ann : 1D-array
|           Disc black-body at annulus - units : W/Hz
|
|   do_nonrelDiscSpec(self)
|       Calculates contribution from entire disc section - for non-relativisitc
|       case. Usefull for comparison...
```

```
|
|     Returns
|     -------
|     Lnu_disc_norel : array
|         Total NON-RELATIVISTIC spectum from standard disc region
|         Units ; W/Hz
|
|  do_nonrelHotCompSpec(self)
|     Calculates spectrum of hot comptonised region - no relativity
|
|     Returns
|     -------
|     Lnu_hot_norel : array
|         Total NON-RELATIVISTIC spectrum from hot Compton region
|         Units : W/Hz
|
|  do_nonrelWarmCompSpec(self)
|     Calculates contribution from entire warm Compton region - for
|     non-relativistic case
|
|     Returns
|     -------
|     Lnu_warm_norel : array
|         Total NON-RELATIVISTIC spectrum from warm Compton region
|         Units : W/Hz
|
|  do_relDiscSpec(self)
|     Calculates contribution from entire disc section - for relativistic
|     case
|
|     Returns
|     -------
|     Lnu_disc_rel : array
|         Total RELATIVISTIC spectrum for standard disc region
|         Units : W/Hz
|
|  do_relHotCompSpec(self)
|     Calculates spectrum of hot compton region - with relativity!
|
|     Returns
|     -------
|     Lnu_hot_rel : array
|         Total RELATIVISTIC spectrum from hot Compton region - units: W/Hz
|
|  do_relWarmCompSpec(self)
|     Calculates contribution from entire warm Compton region - for
|     relativistic case
|
|     Returns
```

```
|       --------
|       Lnu_warm_rel : array
|           Total RELATIVISTIC spectrum from hot Compton region
|           Units : W/Hz
|
|   get_DiscComponent(self, rel=True)
|       Extracts disc component from SED
|
|       First checks if disc region exists in current geometry - if not then
|       returns 0 array
|
|       Parameters
|       ----------
|       rel : Bool, optional
|           Flag for whether to include relativistic correction.
|               - True: Full GR is used
|               - False: Relativistic transfer to the observer is ignored
|                        (i.e non-relativistic SED)
|           The default is True.
|
|       Returns
|       --------
|       Ld : array
|           Disc spectral component in whatever units are currently set.
|
|   get_HotComponent(self, rel=True)
|       Extracts hot Comptonised component from SED
|
|       First checks if hot Compton region exists in current geometry - if not
|       then returns 0 array
|
|       Parameters
|       ----------
|       rel : Bool, optional
|           Flag for whether to include relativistic correction.
|               - True: Full GR is used
|               - False: Relativistic transfer to the observer is ignored
|                        (i.e non-relativistic SED)
|           The default is True.
|
|       Returns
|       --------
|       Lh : array
|           Hot Compton spectral component in whatever units are currently set.
|
|   get_Ledd(self)
|       Gives system Eddington luminosity in whatever units are currently set
|
|       Ignores flux flag - ALWAYS as luminosity
```

```
 |
 |      Returns
 |      -------
 |      Ledd : float
 |          Eddington luminosity.
 |
 |  get_Mdot(self)
 |      Gives PHYSICAL mass accretion rate
 |
 |      If units are: cgs, cgs_wave, or counts - then returns in g/s
 |      If units are: SI or SI_wave - then returns in kg/s
 |
 |      Returns
 |      -------
 |      Mdot : float
 |          Physical mass accretion rate.
 |
 |  get_Rg(self)
 |      Gives scale of one gravitational radius
 |
 |      If units are: cgs, cgs_wave, or counts - then returns in cm
 |      If units are: SI or SI_wave - then returns in m
 |
 |      Returns
 |      -------
 |      R_G : float
 |          Gravitational radius.
 |
 |  get_WarmComponent(self, rel=True)
 |      Extracts warm Comptonised component from SED
 |
 |      First checks if warm Compton region exists in current geometry - if not
 |      then returns 0 array
 |
 |      Parameters
 |      ----------
 |      rel : Bool, optional
 |          Flag for whether to include relativistic correction.
 |              - True: Full GR is used
 |              - False: Relativistic transfer to the observer is ignored
 |                       (i.e non-relativistic SED)
 |          The default is True.
 |
 |      Returns
 |      -------
 |      Lw : array
 |          Warm Compton spectral component in whatever units are currently set.
 |
 |  get_totSED(self, rel=True)
```

```
 |      Extracts the total SED (i.e Ldisc + Lwarm + Lhot)
 |
 |      Parameters
 |      ----------
 |      rel : Bool, optional
 |          Flag for whether to include relativistic correction.
 |              - True: Full GR is used
 |              - False: Relativistic transfer to the observer is ignored
 |                      (i.e non-relativistic SED)
 |          The default is True.
 |
 |      Returns
 |      -------
 |      Ltot : array
 |          Total SED in whatever units are currently set.
 |
 |  hotComp_annuli(self, r, dr)
 |      Calculates spectrum from radial slice of hot comp region
 |      Neccessary in order to apply kyconv correctly!
 |
 |      Note - this is in FRAME of the BLACK HOLE!
 |
 |      Returns
 |      -------
 |      Lnu_ann : array
 |          Spectrum from annular slice of hot Compton region - units : W/Hz
 |
 |  hotCorona_lumin(self)
 |      Calculates the coronal luminosity - used as normalisaton for the
 |      hot compton spectral component.
 |
 |      Calculated as Lhot = Ldiss + Lseed
 |
 |      where Ldiss is the energy dissipated from the accretion flow, and
 |      Lseed is the seed photon luminosity intercpted by the corona
 |
 |      Returns
 |      -------
 |      Lhot : float
 |          Total luminosity of hot Comtpon corona - units : W
 |
 |  new_ear(self, new_es)
 |      Defines new energy grid if necessary
 |
 |      Parameters
 |      ----------
 |      ear : 1D-array
 |          New energy grid - units : keV.
 |
```

```
|  seed_tempHot(self)
|      Calculated seed photon temperature for the hot compton region.
|      Follows xspec model agnsed, from Kubota & Done (2018)
|
|      Returns
|      -------
|      kT_seed : float
|          Seed photon temperature for hot compton - units : keV
|
|  set_flux(self)
|      Sets default output as a flux
|      This ONLY affects spectra! Things like Eddington luminosity, or
|      Bolometric luminosity remain as Luminosity!!
|
|      Note: This will also take the redshift into account!!
|
|      /cm^2 IF cgs or counts, /m^2 if SI
|
|  set_lum(self)
|      Sets defualt output as luminosity (only necessary IF previously set
|      as flux)
|
|  set_units(self, new_unit='cgs')
|      Re-sets default units. ONLY affects attributes extracted through the
|      getter methods
|
|      Note, the only difference between setting cgs vs counts is in spectra
|      Any integrated luminosities (e.g. Ledd) will be given in
|      erg/s IF counts is  set
|
|      Parameters
|      ----------
|      new_unit : {'cgs','cgs_wave', 'SI', 'counts'}, optional
|          The default unit to use. The default is 'cgs'.
|          NOTE, the main cgs_wave will give spectra in erg/s/Angstrom,
|          while cgs gives in erg/s/Hz
|
|  warmComp_annuli(self, r, dr)
|      Calculates comptonised spectrum for annulus at r with width dr. Note,
|      r taken to be in center of bin!
|      Uses pyNTHCOMP
|
|      Parameters
|      ----------
|      r : float
|          Inner radius of annulus - units : Rg.
|      dr : float
|          Width of annulus - units : Rg.
|
```

```
 |      Returns
 |      -------
 |      Lnu_ann : 1D-array
 |          Warm Comptonised spectrum at annulus - units : W/Hz
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  A = 0.3
 |
 |  Emax = 10000.0
 |
 |  Emin = 0.0001
 |
 |  as_flux = False
 |
 |  default_units = 'cgs'
 |
 |  dr_dex = 50
 |
 |  mu = 0.55
 |
 |  numE = 2000
 |
 |  units = 'cgs'
```

1.3.2 Inititating the model - Worked example

2 RELQSO

2.1 Input Parameters

2.2 Running through XSPEC

2.2.1 Installation and Compilation

2.2.2 Fitting to data - Worked example

2.3 Running through PYTHON

2.3.1 Inititating the model - Worked example

2.3.2 Class methods