XNS

Release 4.0.0

N.Bucciantini, J.Soldateschi, A.Pili, L.Del Zanna, K.Franceschetti

Jan 31, 2024

CONTENTS

1 Compiling XNS **3**

2 User parameters **5**
2.1 Convergence . . . 5
2.2 Grid . . . 7
2.3 Printing outputs and files . . . 7
2.4 Rotation . . . 8
2.5 Magnetic field . . . 8
2.6 Initial conditions . . . 9
2.7 Equation of state . . . 9
2.8 Scalar field . . . 10
2.9 Other . . . 10

3 Arrays **11**
3.1 Grid . . . 11
3.2 Metric . . . 11
3.3 MHD . . . 11
3.4 Scalar field . . . 12
3.5 Source terms . . . 12

4 Files and outputs **13**
4.1 Files . . . 13
4.2 Outputs . . . 16
4.3 Visualisation . . . 17
4.4 Reformatting the Equation of State . . . 17

5 Reformatting the Equation os State **19**

6 Examples in GR **21**
6.1 Non-rotating NS in GR with the POL2 EoS and a purely poloidal magnetic field . . . 21
6.2 Non-rotating NS in GR with the POL2 EoS and a purely toroidal magnetic field . . . 28
6.3 Non-rotating NS in GR with the POL2 EoS and a twisted magnetosphere . . . 35
6.4 Uniformly Rotating, unmagnetised NS in GR with the Pol2 EoS . . . 36
6.5 Differnetially Rotating, unmagnetised NS in GR with the APR EoS . . . 36
6.6 Differentially Rotating, toroidal magnetised NS in GR with the Pol2 EoS . . . 36
6.7 Uniformly Rotating, poloidal magnetised NS in GR with the Pol2 EoS . . . 36

7 Examples in STTs **37**
7.1 Non-rotating, unmagnetised NS in STT with the APR EoS . . . 37
7.2 Non-rotating NS in STT with the APR EoS and a purely poloidal magnetic field . . . 44
7.3 Non-rotating NS in STT with the APR EoS and a purely toroidal magnetic field . . . 53

7.4 Non-rotating, unmagnetised NS in STT with the POL2 EoS 62
7.5 Non-rotating NS in STT with the POL2 EoS and a purely poloidal magnetic field 69
7.6 Non-rotating NS in STT with the POL2 EoS and a purely toroidal magnetic field 78

8 Indices and tables **89**

The XNS code solves for axisymmetric equilibria of polytropic magnetized and/or rotating neutron stars (NSs) using the extended conformally flat condition (XCFC) for the metric, in spherical coordinates. This is based on the metric module and the routines developed for the X-ECHO code for GRMHD in dynamical spacetimes (Bucciantini & Del Zanna 2011), which in turn is an upgrade of the Eulerian conservative high-order code (ECHO Del Zanna et al. 2007) for GRMHD in a static background metric (the so-called Cowling approximation). Like ECHO and X-ECHO, also XNS is written in the Fortran90 programming language. The reader is referred to the above cited paper for full derivation of the GRMHD equations, and for the full description of the XCFC solvers.

The 4.0 version of XNS adds the possibility of solving for the structure of a magnetised, rotating NS in a class of theories of gravity alternative to general relativity (GR), that is massless scalar tensor theories (STTs). Moreover, support for the use of realistic, tabulated equations of state (EoS) has been added.

You can download it via Github at (link)

This guide is based on the following papers, where the equations describing the approach for magnetized and rotating models are fully presented:

- Bucciantini N., & Del Zanna L., 2011, A&A, 528, A101 (link).
- Pili A.G., Bucciantini N., & Del Zanna L., 2014, MNRAS, 439, 3541 (link).
- Pili A.G., Bucciantini N., & Del Zanna L., 2015, MNRAS, 447, 2821 (link).
- Pili A.G., Bucciantini N., & Del Zanna L., 2017, MNRAS, 470, 2469 (link).
- Soldateschi, J., Bucciantini, N., & Del Zanna, L. 2020, A&A, 640, A44 (hereafter SBD20) (link).
- Soldateschi, J., Bucciantini, N., & Del Zanna, L. 2021, A&A, 645, A39 (link).
- Soldateschi, J., Bucciantini, N., & Del Zanna, L. 2021, A&A, 654, A162 (link).
- Franceschetti, K., Del Zanna, L., Soldateschi, J., & Bucciantini, N., 2022, Universe, 8, 172 (link).

If you use this software please reference at least one of the previous papers.

CHAPTER

ONE

COMPILING XNS

XNS can be compiled using gfortran in several ways, as written in the file *makefile*:

- make serial - the standard way to compile XNS. In this case a single solution is found with a central density specified by the parameter RHOINI. The generated executable is named *XNS-s*. To compile the code this way type:

```
make clean; make serial; ./XNS-s
```

- make nwtrps - compiles XNS using the Newton-Raphson method to converge on the quantity specified by the parameters QUOC and QUCONV. The generated executable is named *XNS-nr*. To compile the code this way type:

```
make clean; make nwtrps; ./XNS-nr
```

- make parspace - compiles XNS various times with different initial conditions, in order to sample the parameter space spanned by the magnetic, rotation and density parameters, set in *XNS.f90*. In particular, solutions are computed with: NKB different magnetic coefficients spanning from KBMIN to KBMAX; NRHO1 different central densities spanning from RHOMIN to RHOMAX; NOMG different central angular velocities spanning from OMGMIN to OMGMAX. The generated executable is named *XNS-mpi*. The computation of the various models is made in parallel (only on CPUs, no GPU support) using the MPI framework, and mpif90 needs to be installed. To compile the code this way using NUMBER_OF_PROCESSES processes, type:

```
make clean; make parspace; mpirun -n NUMBER_OF_PROCESSES ./XNS-mpi
```

  Note that it must be NUMBER_OF_PROCESSES$\geq$2, because one process is always only passing initial conditions to the other processes, and NUMBER_OF_PROCESSES-1$\leq$(NOMG+1)$\times$(NRHO1+1)$\times$(NKB+1), that is the number of computing processes must not be larger than the number of models to be computed.

- make clean - removes all *.o* and *.mod* files and all executables.
- make cleanall - removes all *.o* and *.mod* files, all executables as well as all *.dat* files.

CHAPTER

TWO

USER PARAMETERS

The input parameters that the user might want to change are all set in the module system inside the file *SYSTEM.f90*. There are other parameters in other parts of the code that deal with specific routines (root finding, convergence etc . . .), but those should not need to be changed. Here is a list of the parameters of the model as they appear in the module system:

2.1 Convergence

2.1.1 Parameters for convergence of the Newton-Raphson scheme of XNS

- **NVALUE** - the maximum number of loops employable by the Newton-Raphson scheme in the search for a equilibrium solution, having a target value for a desired quantity (central density, total mass, etc..) by the program XNS. Usually convergence is reached within about 10 steps, unless the NS is strongly distorted (fast rotation, and/or strong magnetic field). The default value is set to 100.
- **CONVF** - a convergence parameter for the Newton-Raphson scheme in XNS. It is given in relative terms (beware that the code accuracy is $\sim 10^{-3}$).

2.1.2 Parameters for convergence of XNSMAIN over a model

- **MAXLOOP** - the maximum number of loops employable in the search for a converged equilibrium solution by the subroutine XNSMAIN. Usually convergence is reached within the first 100 steps, unless the NS is strongly distorted (fast rotation, and/or strong magnetic field). The default value is set to 100.
- **CONVRHO** - sets if convergence by XNSMAIN should be checked on central density or on central lapse function. CONVRHO should be set to .FALSE. if the central density is held fixed, as in the case of Tabulated EoS
- **OSCCONV** - desired precision in case the solution is reached with oscillatory convergence in XNSMAIN. In this case the check on the convergence is done between a step and the previous few, in such a way to bound the oscillation range.
- **MONCONV** - desired precision in case the solution is reached with monotonous convergence in XNSMAIN. In this case the check on the convergence is done between a step and the previous.

2.1.3 Parameters for convergence of the initial TOV solution

- **RELIT** - the maximum number of loops in TOVINIMOD for the convergence of the full TOV solution, including the metric-matter distribution and, in case of STT, the scalar field.
- **CONVT** - desired precision for the convergence of the full TOV solution and also of the scalar field in TOVINIMOD, in STT.
- **CONV** - desired precision for the convergence of the GR TOV part (or in STT the matter-metric part at fixed scalar field) in TOVINIMOD.
- **MAXSTEPTV** - the maximum number of loops in TOVINIMOD for the convergence of the TOV system to a solution with a fixed scalar field. Note that the full TOV solution is obtained when also the scalar field has reached convergence together with the rest of the quantities.
- **MAXSTEPCH** - the maximum number of loops in TOVINIMOD for the convergence of the scalar field equation of the TOV system to a solution with a fixed metric and matter distribution. Note that the full TOV solution is obtained when also the scalar field has reached convergence together with the rest of the quantities.

2.1.4 Parameters for convergence of the Bernoulli in HYDROEQ

- **CONVHELP** - a logical flag to activate an option in HYDROEQ to help achieve convergence. Warning: if set to .TRUE., the final central density will be slightly different from the chosen RHOINI.

2.1.5 Parameters for the convergence of elliptic solvers

- **TOLCONV** - the convergence tolerance for the iterative solution of the PDEs for the conformal factor ψ and the lapse α in XNSMAIN and for the 4-potential in HYDROEQ.
- **TOLCHI** - desired precision for the scalar field iterative solver in XNSMAIN.

2.1.6 Parameters to help or stabilize convergence

- **QFACTOR** - a damping factor of the convergence loop both for solving the Bernoulli equation, used in HYDROEQ. At the end of each sub-loop of the convergence scheme, a new set of equilibrium fluid variables is computed. Setting QFACTOR=1 implies that these will be used, while setting QFACTOR=0 means that the old variables $V_{\rm old}$ will be used (the code will never converge in this case!). A value $0 < Q_{\rm f} < 1$ implies that at the beginning of each loop a combination of new and old variables will be used, in the form $V = Q_{\rm f} V_{\rm new} + (1 - Q_{\rm f}) V_{\rm old}$. Using a value less than 1 tends to give slower but more stable convergence. Values QFACTOR< 0.5 are to be used only for pathological cases where the convergence is very slow or when the code fails to converge (i.e. rotating models on the unstable branch of NS mass-radius curve).
- **QAPHI** - a damping factor of the convergence loop for solving the Grad-Shafranov or the Maxwell-Ampere equation. Analogous to QFACTOR but for the ϕ-component of the 4-potential.
- **QRELAX** - a damping factor for the convergence loop of the scalar field in TOVINIMOD.
- **QFACTORMETRIC** - a damping factor for the internal convergence loop for solving the conformal factor equation in XNSMAIN.
- **QFACTORCONF** - a damping factor for the conformal factor in the convergence loop of XNSMAIN.
- **QFACTORCHI** - a damping factor for the internal convergence loop for solving the scalar field equation in XNSMAIN.
- **EPS** - a tolerance value. It is used in several subroutines and must be a small value. This should not need to be changed.

2.2 Grid

- **STRETCH** - a logical flag that control whether the grid is stretched or not. If .TRUE. the radial grid is regular up to RREG with NRREG grid points and it is stretched from RREG to RMAXSTR with NR-NRREG points. The stretching factor STRR is determined by the code determined by the code consistently with the choices for NR, NRREG,RMAXSTR and RREG. See also Pili et al. (2015) for details.
- **NR** - the number of radial grid points. The radial grid is defined from r =RMIN and r =RMAX (r =RMAXSTR) is the grid is uniform (stretched).
- **NTH** - the number of angular grid points. The angular grid is always defined between $\theta = 0$ and $\theta = \pi$.
- **NRREG** - number of grid points for the regular grid if STRETCH=.TRUE..
- **RMIN** - the lower boundary in the radial direction. It must be always set to 0, since the metric solver requires a compact domain and has been implemented with specific boundary conditions for RMIN = 0.
- **RMAX** - the maximum radius of the computational domain if the grid is uniform. This can be arbitrarily chosen. However, one needs to guarantee that the NS is properly resolved over a sufficient number of grid points (50-100), so this parameter and NR should be chosen consistently. In particular, the condition RMAX$> 2R_{\rm TOV}$ must hold, where $R_{\rm TOV}$ is the NS radius of the initial TOV guess. This is because the TOV solver is designed to converge when the ADM masses measured at RMAX and RMAX/2 (hence it must be outside the NS) coincide within a given tolerance. If not the code will halt with a warning.
- **RMAXSTR** - the maximum radius of the computational domain if the grid is stretched.
- **RREG** - maximum radius of the regular grid.
- **REQMAX** - the maximum radius beyond which any NS model will be artificially truncated. This must be set $\geq$ of the shedding mass limit. Sometimes, when working with configurations close to mass shedding, during the convergence loop the code might get unbounded solutions or fail to converge. To avoid this, setting a value for REQMAX will force the solution to be truncated.
- **MINRESREG** - desired minimum resolution of the grid, if uniform. In case the resolution is too rough, it allows to issue a warning.
- **MINRESSTR** - desired minimum resolution of the regular part of the grid, if stretched. In case the resolution is too rough, it allows to issue a warning.
- **RINI** - a very small radius used for the expansion of the TOV equations.

2.3 Printing outputs and files

- **VERBOSE** - a logical flag. Setting VERBOSE=.TRUE. forces the code to output on screen all the INFOs related to the various steps done by each subroutine (to be used only for debugging or checks). Otherwise setting VERBOSE=.FALSE. the output on screen will be produced only at the end. The latter is the default option.
- **DEBUG** - additional logical flag used to print more outputs. Used for debugging purposes.
- **WRT** - a logical flag. Setting WRT=.TRUE. forces the code and each subroutine to write output files at every step or substep, otherwise setting WRT=.FALSE. will prevent IO writing.
- **WRTF** - a logical flag. Setting WRTF=.TRUE. override WRT for the final step, and allow to write all the files related to the final configuration. Setting WRTF=.FALSE. will prevent IO writing.
- **CHUP** - a logical flag. Setting CHUP=.TRUE. allows (subject to WRT, WRTF) to write the files containing the results of the metric solver and primitive solver XShiftphi.dat, Conformal.dat, Primitive.dat, Primitive_mag.dat, Shiftphi.dat, Lapse.dat, Source.dat, Rhovec.dat, Chi.dat. Setting CHUP=.FALSE. will prevent from writing these files. The latter is the default option, unless one wishes to perform a check of the metric or primitive solvers.

- **WGRID** - a logical flag. Setting WGRID=.TRUE. writes the grid in a file named Grid.dat.
- **WCONVA** - a logical flag. Setting WCONVA=.TRUE. writes two files named Apconv.dat and Atconv.dat used to check the convergence of the t and ϕ components of the 4-potential.
- **WCONVC** - a logical flag. Setting WCONVC=.TRUE. writes a file named Chiconv.dat used to check the convergence of the scalar field.

2.4 Rotation

Warning: rotation in concurrency with the presence of a scalar field has not been tested!

- **OMG** - the value of the angular velocity at the center Ω_c.
- **A2VALUE** - the value of A2. This is needed only for differentially rotating models, otherwise it should be set to 0.
- **DIFFERENTIAL** - a logical flag that states whether the model is differentially rotating or not. Setting it to .FALSE. implies uniform rotation, with Ω =OMG. Setting it to .TRUE. implies differential rotation. In this case, a value of A2VALUE must be specified.
- **OMGSPACE** - Logical variable to work in Ω or J space. If .TRUE. the rotational law has form $J(\Omega)$, else $\Omega(J)$.
- **JCONSTLAW** - Rotation law: $J = A^2(\Omega_c - \Omega)$
- **JCMODLAW** - Rotation law: $J = A^2\Omega[(\Omega_c/\Omega)^p - 1]$
- **URYULAW3** - Rotation law: Uryu 3, $\Omega = \Omega_c\,[1 + (\frac{j}{B^2\,\Omega_c})^p](1 - \frac{j}{A^2\,\Omega_c})$
- **URYULAW4** - Rotation law: Uryu 4, $\Omega = \Omega_c\,\frac{1 + (j/B^2\,\Omega_c)}{1 + (j/A^2\,\Omega_c)^4}$
- **PROTDIFF** - The index p in either JCMODLAW or URYULAW3.
- **OMGMAX** - The maximum value of the rotation rate for Uryu 3 and 4 (this is given instead of A and B)
- **RMVALUE** - The radius at which the maximum value of the rotation rate for Uryu 3 and 4 is reached (this is given instead of A and B)

2.5 Magnetic field

Warning: a twisted torus configuration in concurrency with the presence of a scalar field has not been tested!

- **IMAG** - a logical flag that states whether the model is magnetized or not. Setting it to .FALSE. implies the non magnetized case. Setting it to .TRUE. implies the presence of a magnetic field. In the latter case, values of parameters for the magnetic model must be specified.
- **ITOR** - a logical flag that must be set true only for purely toroidal configurations.
- **IPOL** - a logical flag that must be set true only for purely poloidal configurations.
- **ITWT** - a logical flag that must be set true only for mixed Twisted Torus configurations.
- **BCOEF** - the value of Km in the magnetic polytropic law for the case of purely toroidal field. Never used when IMAG=.FALSE. or ITOR=.FALSE., though it is better set to 0 in this case.
- **MAGIND** - the value of m in the magnetic polytropic law for the case of purely toroidal field. It must be > 1, otherwise the magnetic energy diverges on the polar axis. It is never used when IMAG=.FALSE. or ITOR=.FALSE..

- **KBPOL** - the value of $K_{\rm pol}$ in the magnetic law for the case of purely poloidal field. Never used when IMAG=.FALSE. or IPOL=.FALSE., though it is better set to 0 in this case.
- **CSI** - the value of the parameter ξ (non linear current term) in the magnetic polytropic law for the case of purely poloidal field. It is never used when IMAG=.FALSE. or IPOL=.FALSE..
- **QNULL** - logical flag that regulates the global net charge of a rotating star with poloidal magnetic field. If QNULL=.TRUE. the code searches for a globally uncharged star, otherwise it minimizes the electric field at the stellar pole.
- **KBTT** - the value of of $K_{\rm pol}$ in the magnetic law for the case of mixed Twisted Torus configuration. Never used when IMAG=.FALSE. or ITWT=.FALSE., though it is better set to 0 in this case.
- **ATWT** - the value of a in the magnetic law for the case of mixed Twisted Torus configuration. Never used when IMAG=.FALSE. or ITWT=.FALSE., though it is better set to 0 in this case.
- **ZETA** - the value of ζ in the magnetic law for the case of mixed Twisted Torus configuration.
- **CUT** - the value of λ in the magnetic law for twisted magnetosphere models. It regulates the extension of the twist. If $\lambda = 1$ standard Twisted Torus models are recovered.
- **NPOL** - value of the magnetic powerlaw index. Currently it is not used.

2.6 Initial conditions

- **RHOINI** - the central density for the starting guess. If the serial flag is used to compile the code, this is also the central density of the final solution. WARNING: if the nwtrps flag is used to compile the code, this is not the central density of the converged model, but its value for the starting guess. By default XNS will search for a solution in the range 0.8-1.2 RHOINI. If the desired solution is outside this range, XNS will output a warning, and stop.
- **QUOC** - the quantity of interest to which the model must converge [0 for a given central density, 1 for a given gravitational mass, 2 for a given barionic mass].
- **QUCONV** - the value of the quantity of interest to which we want a model to converge. For example if one wants a model with central density 1.28×10^{-3} (in geometrized units) set: QUOC=0, QUCONV=1.28E-3.

2.7 Equation of state

- **K1** - the polytropic coefficient K of the EoS. The code uses a polytropic EoS. The value K1=100 is for the standard case used for many tests of NS stability and evolution in the literature (see also the parameter below).
- **GAMMA** - the adiabatic index $\gamma = 1 + 1/n$ of the polytropic EoS, where n is the polytropic index. The value GAMMA=2 (n =1) is for the standard case used for many tests of NS stability and evolution in the literature (see also the parameter above).
- **CTP** - if set to .FALSE. the code avoid to use conservative to primitive routines. This flag is effective only with IPOL=.TRUE..
- **MBARYONFC** - ratio between a reduced baryon mass and true baryon mass. If equal to 1, it has no effect of the code. In case one wants to find a solution using the SQM2 EoS (tabulated in the SQ2_resampled.dat file), it is necessary to use MBARYONFC=0.86.
- **EOSINT** - a logical flag to choose whether o use a tabulated or a polytropic EoS. If EOSINT=.TRUE., the EoS tabulated in the file in FILEEOS is used. Otherwise, a polytropic EoS is used.
- **FILEEOS** - specifies the name of the file containing the tabulated EoS that we want to use.

- **NPTRHO** - number of points of the tabulated EoS.
- **EOSJOR** - a logical flag that specifies in which frame of STTs (either Jordan or Einstein) the EoS is computed. If EOSJOR=.TRUE., it is computed in the Jordan frame; otherwise, in the Einstein frame. It should always be set to .FALSE.. Note that, in GR, the two frames coincide, and so this flag has no effect on a GR solution.

2.8 Scalar field

- **ALPHA0** - value of the α_0 parameter of the scalar coupling function $\mathcal{A}(\chi) = \exp[\alpha_0(\chi-\chi_{\rm inf})+\beta_0(\chi-\chi_{\rm inf})^2]$ by Damour, T., & Esposito-Farèse, G. (1993). Note that, if set to zero, XNS converges to a GR solution.
- **BETA0** - value of the β_0 parameter of the scalar coupling function $\mathcal{A}(\chi)$.
- **CHIINF** - value of the scalar field at infinity $\chi_{\rm inf}$ in the scalar coupling function $\mathcal{A}(\chi)$.

- **GR** - a logical flag used to choose whether to compute the solution in GR (if .TRUE.) or in STTs (if .FALSE.). Note that setting GR=.FALSE. together with ALPHA0=0 and BETA0=0 leads to the GR solution anyway.

2.9 Other

- **RHOSURF** - value at which the surface of the NS is set.
- **MLS** - number of spherical harmonics (Legendre polynomials) for spectral decomposition in θ (numbered from 0 to MLS). This should be <NTH. In 1D (NTH=1) it must be set to 0.
- **NGQ** - the number of interpolation points for the Gauss quadrature, needed to compute the integrals over the polar direction of the source terms in the spherical harmonics decomposition. Used by all the 2D elliptical PDE solvers. It must be NGQ$$NTH. In 1D (NTH=1) it must be set to 1.
- **MLSL** - number of spherical harmonics used to solve the Laplace equation as described in Pili et al. (2017). This parameter is relevant only in the cases of rotating and poloidal magnetized star.
- **ANALYTIC** - a logical flag. If ANALYTIC=.TRUE., the derivatives of the scalar field in TOVINIMOD are computed using their analytic form; otherwise, a numerical derivative is computed. This option may be used to achieve better convergence of the TOV solution in case one approach is numerically unstable.
- **DELTMU0** - variation $\Delta\mu_0$ of the central value μ_0 of μ (where $-e^{\mu} = g_{00}$ is the 00 element of the metric tensor in the TOV metric). It is used to compute the solution at a fixed scalar field in the Newton-Raphson method in TOVINIMOD. Its value should not be changed.
- **MUIN** - initial guess on μ_0 in the TOV solution.
- **MMID** - initial guess on the ADM mass of the TOV solution computed in the middle of the grid. At convergence, it must be M∼MMID.
- **M** - initial guess on the ADM mass of the TOV solution computed at the outer edge of the grid. At convergence, it must be M∼MMID.

CHAPTER

THREE

ARRAYS

The module system inside the file *SYSTEMXNS.f90* also contains the definitions of some arrays that are used within the code and shared by many subroutines. We briefly describe some of them here so that the user can have some idea of what represents what. Other arrays that are specific only to certain subroutines are defined locally and are not discussed here. Notice that some arrays related to the poloidal components of the velocity, shift vector, or auxiliary vectors are always zero but still defined in XNS, though related routines are never called. This is because XNS shares the same metric solver as the full X-ECHO code.

3.1 Grid

- **R**,**DR** - 1D arrays that store the location of the radial points, and the radial increments.
- **DRM**,**DRP** - additional 1D arrays that store the backward and forward radial increments.
- **TH**,**DTH**,**XX** - 1D arrays that store the location of the angular points, the angular increments, and the cosine of the angle.

3.2 Metric

- **PSI**,**PSL**,**PSS**,**PSSR**,**PSST** - 2D arrays of metric terms, respectively ψ, $\alpha\psi$, X^ϕ, X^r, X^θ. We have either $X^i = W^i$, or $X^i = \beta^i$, depending on the step of the metric solver.
- **CURVC**,**CURVR**,**CURVT**,**CURVP** - 2D arrays containing the source terms associated with the curvature of the metric, respectively for the two scalar Poisson equations (for ψ and α) and for the three components of the second vector Poisson equation (that for β^i).
- **MU**,**NU** - 1D arrays containing the metric terms of the radial TOV solution. The metric employed is that for isotropic coordinates, namely $ds^2 = -e^\nu dt^2 + e^\mu(dr^2 + r^2 d\theta^2 + r^2 \sin^2\theta d\phi^2)$.

3.3 MHD

- **RHOSRC**,**ESRC**,**PSRC**,**VPHI**,**VR**,**VTH**,**BPHI**,**SSS** - 2D arrays, respectively ρ, $\rho h = \rho(1+\varepsilon) + p$, p, v^ϕ, v^r, v^θ, B^ϕ, S, needed for the source terms.
- **RHOTV**,**PRTV**,**ETV** - 1D arrays containing the fluid variables of the radial TOV solution, respectively ρ, p, $\rho\varepsilon$.
- **RHONEW**,**PNEW**,**ENEW**,**V3NEW**,**B3NEW**,**E3NEW** - 2D arrays, respectively ρ, p, $\rho\varepsilon$, v^ϕ, B^ϕ, E_ϕ computed for an equilibrium configuration on the metric at the end of each step of the convergence loop.

- **BPOLR**,**BPOLT**,**EPOLR**,**EPOLT**,**APHI**,**ATIM** - 2D arrays, respectively, B^r, B^θ, E_r, E_θ, $\tilde{A}^\phi = \Phi$, $A^t = \Psi$ for the magnetic configuration with poloidal field components.
- **RHOTVJOR**,**PRTVJOR**,**ETVJOR** - 1D arrays, respectively ρ, p, $\rho\varepsilon$ of the TOV solution computed in the Jordan frame. Note that, in GR, these are equal to RHOTV, PRTV, ETV respectively.

3.4 Scalar field

- **CHITV**,**DCHITV**,**DDCHITV** - 1D arrays, respectively χ, $\partial_r\chi$, $\partial_r^2\chi$ of the TOV solution.
- **CHI** - 2D array containing the scalar field χ.
- **PSCAL**,**QSCALTIM**,**QSCALR**,**QSCALT**,**QSCALP**,**QSCAL2** - 2D arrays, containing the 3+1 decomposition of the scalar field derivative. These are respectively P, Q^μ and $Q_\mu Q^\mu$ of Eq. 26 in SBD20.
- **ASCAL** - 2D array, containing the scalar coupling function $\mathcal{A}(\chi)$.

3.5 Source terms

- **USRC**,**DSRC**,**S3SRC**,**S1SRC**,**S2SRC** - 2D arrays containing the U conservative variables needed for the source terms, respectively $\hat{E}$, $\hat{D}$, $\hat{S}_\phi$, $\hat{S}_r$, $\hat{S}_\theta$, all multiplied by $f^{1/2} = r^2 \sin\theta$.
- **USRCX**,**S3SRCX**,**S1SRCX**,**S2SRCX** - 2D arrays containing the U conservative variables associated to the scalar field, needed for the source terms, respectively $\hat{E}$, $\hat{D}$, $\hat{S}_\phi$, $\hat{S}_r$, $\hat{S}_\theta$, all multiplied by $f^{1/2} = r^2 \sin\theta$.
- **ECSRC**,**ELSRC**,**ES1RC**,**ES2RC**,**ES3RC** - 2D arrays containing the source terms (the right hand side of the equations) associated with the presence of matter in the elliptic PDEs. Respectively, the source for the equations for ψ, $\alpha\psi$, X^ϕ, X^r, X^θ, where $X^i = W^i$, or $X^i = \beta^i$.
- **TRACEM** - 2D array containing the trace of the energy-momentum tensor of matter, used for the source of the equation of the scalar field.

CHAPTER

FOUR

FILES AND OUTPUTS

Here is a list of all the files and subroutines included int the XNS package, together with a brief description of what they do and how they operates. We start with the Fortran 90 files of the code (with extension *.f90*), then we describe the output files produced by a run (with extension *.dat*), and we conclude with the Python 3 files needed for visualisation (with extension *.py* or *.ipynb*, for use in Jupyter). The code must run in double precision for convergence. The *makefile* is provided for the gfortran (GNU) compiler. For more info on the compilation of XNS, see section "Compiling XNS".

4.1 Files

- **XNS.f90** - main program. Makes some consistency checks, and invokes XNSMAIN. Depending on the pre-compiling option it simply calls XNSMAIN (if make serial is used to compile the code), or it performs a Newton-Raphson search for an equilibrium model with a given value for a desired quantity of interest, i.e. a certain value of the central density or gravitational mass (if make nwtrps is used). Moreover, it can compute many models in parallel with different initial conditions (if make parspace is used).
- **XNSMAIN.f90**
  - subroutine **XNSMAIN** - the main kernel of the code: it defines the grid, builds a 2D initial guess based on the 1D TOV output of TOVINIMOD.f90, performs the convergence loop calling all the various metric solvers and procedures in the appropriate order. When the loop is over, it writes all the outputs.
  - subroutine **CONFORMAL** - solves for the scalar Poisson-like equation for ψ.
  - subroutine **LAPSE** - solves for the scalar Poisson-like equation for $\alpha\psi$.
  - subroutine **SHIFTPHI** - solves the ϕ component of the two vector Poisson equations for W^i and β^i, given the corresponding source terms.
  - subroutine **CURV1** - computes the curvature source term in the routines for ψ and $\alpha\psi$.
  - subroutine **CURV2** - computes the curvature source term in the routine for β^ϕ.
  - subroutine **SOURCECHI** - computes the metric terms and derivatives used in the source terms of the equation for χ.
  - subroutine **SOLVECHI** - solves for the scalar field equation for χ, given the source term. Note that since χ also appears in the source term, a single call of SOLVECHI won't yield a true solution.
  - subroutine **CHISOL** - wrapper that calls SOURCECHI and CHISOL in a relaxation loop, until it converges to the true solution.
  - subroutine **CHIDERIVS** - computes the derivatives of the scalar field (Q^μ) and the scalar coupling function $\mathcal{A}(\chi)$.
- **SYSTEM.f90**

- module **SYSTEM** - contains various parameters of the run, to be specified by the user, and definitions of common arrays (see section "User parameters").
- subroutine **EOSTABLEREAD** - reads the EoS file specified by the FILEEOS parameter. Note that the first line in the file must be the number of points present in the file, and this must be equal to the parameter NPTRHO. The subsequent lines must contain the minimum and maximum density and their indexes (second line), the minimum and maximum pressure and their indexes (third line), the minimum and maximum internal energy and their indexes (fourth line), the minimum and maximum enthalpy and their indexes (fifth line). Then, the table is read. Please, refer to the routine code to see the specific structure that the EoS file must have.
- subroutine **RHO2EOS** - given ρ, it computes the pressure p, the internal energy ε and the enthalpy h according to the tabulated EoS.
- subroutine **PRS2EOS** - given p, it computes the ρ according to the tabulated EoS.
- subroutine **ENT2EOS** - given h, it computes the ρ according to the tabulated EoS.
- subroutine **EOS** - computes the density and the internal energy given the pressure, both in case the EoS is tabulated or an analytical polytropic.
- subroutine **FUNCD_EOS** - used by the root-finding subroutine to derive the central pressure given the central density.

• **HYDROEQ.f90**

- subroutine **HYDROEQ** - given the CFC metric and a value of $\rho_{\rm c}$ it computes the equilibrium configuration for the corresponding Bernoulli integral. It finally calls hydrovar_ depending on the physical parameter set in SYSTEMXNS.f90 to compute local equilibrium quantities.
- subroutine **HYDROVAR**, **HYDROVAR_TOR**, **HYDROVAR_POL** - they compute local equilibrium quantities such as ρ, p, v^ϕ, B^i and E_i depending on the specific choice for the magnetization (respectively unmagnetized case, purely toroidal magnetic field and poloidal magnetic field).
- subroutine **COVTERM** - computes the local terms of the metric tensor
- subroutine **CONS_TO_PRIM** - computes the inversion from conserved to primitive variables.
- subroutine **CONS_TO_PRIM_POL** - computes the inversion from conserved to primitive variables for the specific case of poloidal field.
- subroutine **QUANTITIES** - computes several quantities (e.g. mass, energy, angular momentum, scalar charge, magnetic deformation) at the end of the convergence loop, according to standard definitions. See also SDB20 for some definitions in the case of STTs.
- subroutine **SOURCEPOT** - compute source terms (currents and metric) for the Grad-Shafranov Equation or Maxwell equations depending if the rotational rate OMG is set to zero or not.
- subroutine **VECPOTPHI** - called by the subroutine hydrovar_pol when OMG.EQ.0, it solves the Grad-Shafranov Equation.
- subroutine **MXWLSOL** - called by subroutine hydrovar_pol when OMG.NE.0, it solves iteratively the Maxwell-Ampere and the Maxwell-Gauss equation. It finally corrects the solution for the electric potential Φ in order to guarantee that the MHD condition $\Phi = -\Omega\Psi + C$ is valid inside the star. Indeed, as explained in Pili et al. (2017), the solution for Φ obtained by solving the non-homogeneus Maxwell equations, does not satisfy the perfect conducting relation inside the star, but differs from the MHD solution solution $\Phi_{\rm MHD} = -\Omega\Psi + C$ by an harmonic function $\Phi_{\rm a}$ so that $\Phi = \Phi_{\rm MHD} + \Phi_{\rm a}$ with $\Delta\Phi_{\rm a} = 0$. The harmonic function is obtained evoking the laplace subroutine.
- subroutine **SOLVEAPHI** and subroutine **SOLVEATIME** - solve respectively for the Maxwell-Ampere and Maxwell-Gauss equations.

  - subroutine **LAPLACE** - solves the equations $\Phi_{\rm a}\big|_{S_{\rm NS}} = \Sigma_l Y(\theta)(a_l r^l)|_{S_{\rm NS}}$ (inside the star) and $\Phi_{\rm a}\big|_{S_{\rm NS}} = \Sigma_l Y(\theta)(b_l r^{-(l+1)})|_{S_{\rm NS}}$ (outside the star), where $S_{\rm NS}$ is stellar surface and $\Phi_{\rm a}\big|_{S_{\rm NS}} = (\Phi + \Omega\Psi + C)|_{S_{\rm NS}}$. Each system of equations is solved with a LU decomposition and a subsequent backward substitution adopting the routines provided in the Numerical Recipes (ludcmp and lubksb). Notice that, in order to avoid spurious effects, the surface terms are evaluated on top of the super-ellipsoid that best fit the numerical surface.

- **ROTATION.f90**
  - subroutine **CHECKROTDIFF** - performs several checks for consistency in the choice of the options and parameters for differential rotation.
  - subroutine **OMEGAVALUE** - derives the function $\Omega = \Omega(r, \theta)$ for the differential rotation.
  - subroutine **OMEGA3LVALUE** - derives the function $\Omega = \Omega(r, \theta)$ and A3L ($\int j d\Omega$ or $\int \Omega dj$, depeding if omega or J-space, the parameter the enters the Bernoulli Equation for the differential rotation) for the differential rotation.
  - subroutine **FODFO_OS** - evaluate FO and DFO required to compute the local rotation rate, in the Omega-space, for a differential rotator.
  - subroutine **A3L_OS** - evaluate A3L $= \int j d\Omega$ in the Omega-space for a differential rotator. This is the parameter the enters the Bernoulli Equation for the differential rotation.
  - subroutine **PARS_VALUE_JS** evaluate JMAX, XVAL and YVAL (only needed for Uryu Laws, see Franceschetti et al. 2022 A8, A9, A18).
  - subroutine **FODFO_JS** - evaluate FO and DFO in the J-space for a differential rotator (only needed for Uryu Laws, see Franceschetti et al. 2022 A2, A3).
  - subroutine **A3L_JS** evaluate A3L $= \int \Omega dj$ in the J-space for a differential rotator. This is the parameter the enters the Bernoulli Equation for the differential rotation.
- **TOVINIMOD.f90**
  - subroutine **TOVINIMOD** - solves the 1D TOV (either in GR or in STTs) equations in isotropic coordinates to provide the initial guess. It uses a relaxation method to achieve convergence.
  - subroutine **EXPANSION** - a Taylor expansion of the TOV equations at small initial radii (they are singular for $r \to 0$).
  - subroutine **TOVEQS** - provides the derivatives needed to integrate the TOV equations via the RK4 method.
  - subroutine **RK4** - the 4th order RK integrator (modified from the Numerical Recipes).
  - subroutine **MASSFIND** - computes the ADM mass of the scalarised TOV solution at a given radius (either at the middle of the grid or at its outer edge) by knowing the scalar charge and the value of μ at that point. It is used in order to achieve convergence, as explained in SBD20 Appendix B.
- **PHYSICS.f90**
  - subroutine **GRIDBUILD** - computes the radial grid (either uniform or stretched) and derivative terms used in the DGTSV subroutine.
  - subroutine **FUNCD_STRETCH** - used by the root-finding subroutine to derive the stretching factor for the grid, if it is stretched.
- **FUNCTIONS.f90**
  - subroutine **DGTSV** - solves the linear system $AX = B$, where A is a tridiagonal matrix, by Gaussian elimination with partial pivoting (taken from the LAPACK routines).
  - subroutine **LUSOLVER** - solves the linear system $AX = B$, where A is an $N \times N$ matrix and B is a vector of length N, with an LU decomposition.

- subroutine **MYSWAP** - exchanges two arrays, adapted from LAPACK 3.2
- function **MYISAMAX** - returns the index of maximum value of an array, adapted from LAPACK 3.2
- subroutine **SGER** - performs the rank 1 operation A := $x \times y^T$ + A, where x is an nn element vector, y is an n element vector and A is an nn by n matrix. Adapted from LAPACK 3.2
- subroutine **SLASWP** - exchange elements of an array, adapted from LAPACK 3.2
- subroutine **STRSM** solve A*X = B where A is a matrix in LU form, adapted from LAPACK 3.2
- subroutine **LAGRANGEINT** - interpolate YP(XP) in XINT using (N-1)th-Polinomial according to Lagrange Interpolation with Neville algorithm
- subroutine **LAGRANGEINT2D** - perform 2D interpolation using (N-1)th-Polinomial according to Lagrange Interpolation with Neville algorithm, on structured grid
- subroutine **LEGZO** - computes the zeros of Legendre polynomials and the corresponding weights for Gaussian quadrature integration. From: SPECIAL_FUNCTIONS by Shanjie Zhang, Jianming Jin.
- subroutine **LPN** - computes the Legendre polynomials and their derivatives. From: SPECIAL_FUNCTIONS by Shanjie Zhang, Jianming Jin.
- subroutine **POLINT** - a polynomial 2nd order interpolation routine (e.g. Numerical Recipes).
- subroutine **XNS2ECHO_OUT** - output file for X-ECHO

4.2 Outputs

- **Grid.dat** - contains the grid mesh points.
- **TOVINIMOD_PROFILES.dat** - contains the 1D TOV solution $(r, \mu, \rho, \nu, p, \rho\varepsilon, \chi)$.
- **Source.dat** - contains 2D source term for the metric solver $(\rho, p, \rho\varepsilon)$.
- **XShiftphi.dat** - contains the W^ϕ component and the related source term of its vector Poisson equation.
- **Conformal.dat** - contains ψ and the two (matter and curvature) source terms of its scalar Poisson equation.
- **Primitive.dat** - contains the primitive variables $(\rho, p, \rho\varepsilon, v^\phi, B^\phi)$ recovered self-consistently from the metric and the conserved variables.
- **Primitive_mag.dat** - contains the magnetic primitive variables (B^ϕ, B^r, B^θ) recovered self-consistently from the metric and the conserved variables.
- **Lapse.dat** - contains α and the two (matter and curvature) source terms of the related scalar Poisson equation.
- **Shiftphi.dat** - β^ϕ vector and the related source term of the vector Poisson equation.
- **Chi.dat** - χ and the related source term of the scalar field equation.
- **Hydroeq.dat** - contains the new equilibrium configuration $(\rho, p, \psi, v^\phi, \alpha, \beta^\phi, \chi, Q^r, Q^t)$.
- **Hydroeq_mag.dat** - contains the new equilibrium configuration for magnetic field $(B^\phi, B^r, B^\theta, \tilde{A}^\phi, E_\phi, E_r, E_\theta, A^t, J^\phi, J^r, J^\theta)$.
- **Mxwll_test.dat** - cointains data related to the source term of both Maxwell-Ampere and Maxwell-Gauss equation $(\rho_\mathrm{e}, J^\phi, \Phi_\mathrm{int}, \Phi_\mathrm{ext}, \Phi_\mathrm{a}, \omega, \Gamma)$.
- **Apconv.dat** - maximum value of Ψ at each step of the XNSMAIN subroutine.
- **Atconv.dat** - maximum value of Φ at each step of the XNSMAIN subroutine.
- **Chiconv.dat** - maximum value of χ at each step of the XNSMAIN subroutine.

- **Rhovec.dat** - central density and other quantities at each step of the XNSMAIN subroutine. It is used to check convergence.
- **Surf.dat** - contains the radius of the NS surface at all angles θ.
- **LogFile.dat** - summary of the run (input and output quantities).

4.3 Visualisation

- **starplot_polo.py** - plots a section of the star in the $x - z$ plane along with the contours of either the poloidal magnetic field (with or without the field lines), the scalar field or the density.
- **starplot_toro.py** - plots a section of the star in the $x - z$ plane along with the contours of either the toroidal magnetic field, the scalar field or the density.
- **starplot_unmag.py** - plots a section of the star in the $x - z$ plane along with the contours of either the scalar field or the density.
- **profile_polo.py** - plots the radial profiles, both at the pole and at the equator, of several quantities: ρ, p, ψ, α, χ, $B_{\rm pol}$.
- **profile_toro.py** - plots the radial profiles, both at the pole and at the equator, of several quantities: ρ, p, ψ, α, χ, $B_{\rm tor}$.
- **profile_unmag.py** - plots the radial profiles, both at the pole and at the equator, of several quantities: ρ, p, ψ, α, χ.

4.4 Reformatting the Equation of State

- **resample.py** - PYTHON script that takes an EoS (typically in COMPSTAR format, but can be any other), and rebuild a table in the XNS 4.0 format.

CHAPTER

FIVE

REFORMATTING THE EQUATION OS STATE

XNS 4.0 can now take tabulated realistic EoS and compute rotating-magnetized GR and STT NS-models. XNS 4.0 uses Geometrized Units with $G = c = M_\odot = 1$. It uses a specific formatted tabulated EoS, that differs from most of available EoSs for various reasons:

- To speed up interpolation, the EoS has been resampled and re-tabulated using uniform spacing in log-space for the input variable. This means that we have a table for $p(\rho)$, $e(\rho)$ etc… and another table for $\rho(p)$, and another one for $\rho(h)$ etc….
- XNS 4.0 uses the pseudo-hentalpy h which is almost never given in tabulated EoSs. So we need to compute it and add to our tabuled formatted EoS. More precisely we tabulate the natural logarithm $Ln(h)$ defined, in terms of the rest mass energy density ρ, internal energy density e and pressure p as: $\mathrm{d}Ln(h) = \mathrm{d}p/(\rho + e + p)$
- XNS 4.0 is a spin-off of X-ECHO, and as such operates using rest mass (energy) density ρ, internal energy density e and pressure p. Normally EoSs are given in terms of the total energy density $e_{\rm tot}$ and the baryon number density $n_{\rm b}$. As a consequence to use XNS 4.0, one needs to split the total energy density between rest mass and internal energy density. We do this setting $e = e_{\rm tot} - n_{\rm b} m_{\rm b}$ where $m_{\rm b}$ is the baryon mass, assumed to be constant. However it can happen that $n_{\rm b} m_{\rm b} > e_{\rm tot}$, either at very low densities (when the two are almost equal) due to round-off and approximation errors, or at very high densities due to quark deconfinement (the rest mass of three deconfined quarks is less than the rest mass of a nucleon in which the same three quarks are confined, even if the baryon number is the same). This means that in recomputing our formatted EoS we must ensure that $n_{\rm b} m_{\rm b} < e_{\rm tot}$ always. For this reason the value of $m_{\rm b}$ can be tuned to avoid such problem. There is a parameter **MBARYONFC** to take care of it (you can find more on the guide). For nuclear EoSs, the problem arises only at low densities and usually a factor ± 1.0005 is sufficient. Being this within the accuracy of the code, one needs only to check the EoS behave properly, without activating the relative option/parameter. For deconfined systems one might need to change it more, and in this case also activate the relative option.

In order to do all of this we have a PYTHON script that takes an EoS (typically in COMPSTAR format, but can easily be adapted to any other, the EoS for the default script are in the directory BL2), and rebuild a table in the XNS 4.0 format. Beware that you need to check that everything works (the code has some check), in particular if you join different EoSs check that they behave properly (it can happen that EoSs computed by different groups, in different density ranges, and with different approaches cannot easily be joined together without introducing artefacts or jumps - these might create problems to XNS 4.0 if not properly treated). Sometimes, even if the input EoS behaves properly, our python script can produce a wrong interpolation. This typically happens when two different EoS are joined (like at the edge of the crust) if the joining is not smooth enough. XNS 4.0 can work even in the presence of phase-transitions and/or density jumps like for Strange-Stars, but in general the presence of strong discontinuities can create issues preventing a proper relaxation.

Format of a typical XNS 4.0 Tabulated EoS.

- Line 1 - number of points used for the tabulated EoS (i.e. between the minimum density and the maximum density) = 1000.
- Line 2 Empty
- Line 3 minimum and maximum $\mathrm{Log}_{10}(\rho)$

- Line 4 minimum and maximum $\mathrm{Log}_{10}(p)$ and the effective adiabatic index $\Gamma_p = \mathrm{d}Log(p)/\mathrm{d}Log(\rho)$ at minimum and maximum pressure.
- Line 5 minimum and maximum $\mathrm{Log}_{10}(e)$ and the effective adiabatic index $\Gamma_e = \mathrm{d}Log(e)/\mathrm{d}Log(\rho)$ at minimum and maximum internal energy.
- Line 6 minimum and maximum $\mathrm{Log}_{10}(Ln(h))$ and the effective adiabatic index $\Gamma_h = \mathrm{d}Log(Ln(h))/\mathrm{d}Log(\rho)$ at minimum and maximum $Ln(h)$.
- Empty line
- Block 1 (1000 lines - 7 columns) - First column is $Log_{10}(\rho)$ uniformly sampled (1000 points) between $Log_{10}(\rho_{min})$ and $Log_{10}(\rho_{max})$. Second column is $Log_{10}(p)$ at the same densities. Third column is $Log_{10}(e)$. Forth column is $Log_{10}(Ln(h))$ at the same density. Fifth column is Γ_p, sixth column is Γ_e, seventh column is Γ_h.
- Empty line
- Block 2 (1000 lines - 3 columns) - First column is $Log_{10}(p)$ uniformly sampled between $Log_{10}(p(\rho_{min}))$ and $Log_{10}(p(\rho_{max}))$. Second column is $Log_{10}(\rho)$ at the same pressure.Third column is $\Gamma_{\rho-p} = \mathrm{d}Log_{10}(\rho)/\mathrm{d}Log_{10}(p)$ at the same pressure.
- Empty line
- Block 3 (1000 lines - 3 columns) - First column is $Log_{10}(Ln(h))$ uniformly sampled between $Log_{10}(Ln(h(\rho_{min})))$ and $Log_{10}(Ln(h(\rho_{max})))$. Second column is $Log_{10}(\rho)$ at the same hentalpy. Third column is $\Gamma_{\rho-h} = \mathrm{d}Log_{10}(\rho)/\mathrm{d}Log_{10}(Ln(h))$ at the same hentalpy.

CHAPTER

SIX

EXAMPLES IN GR

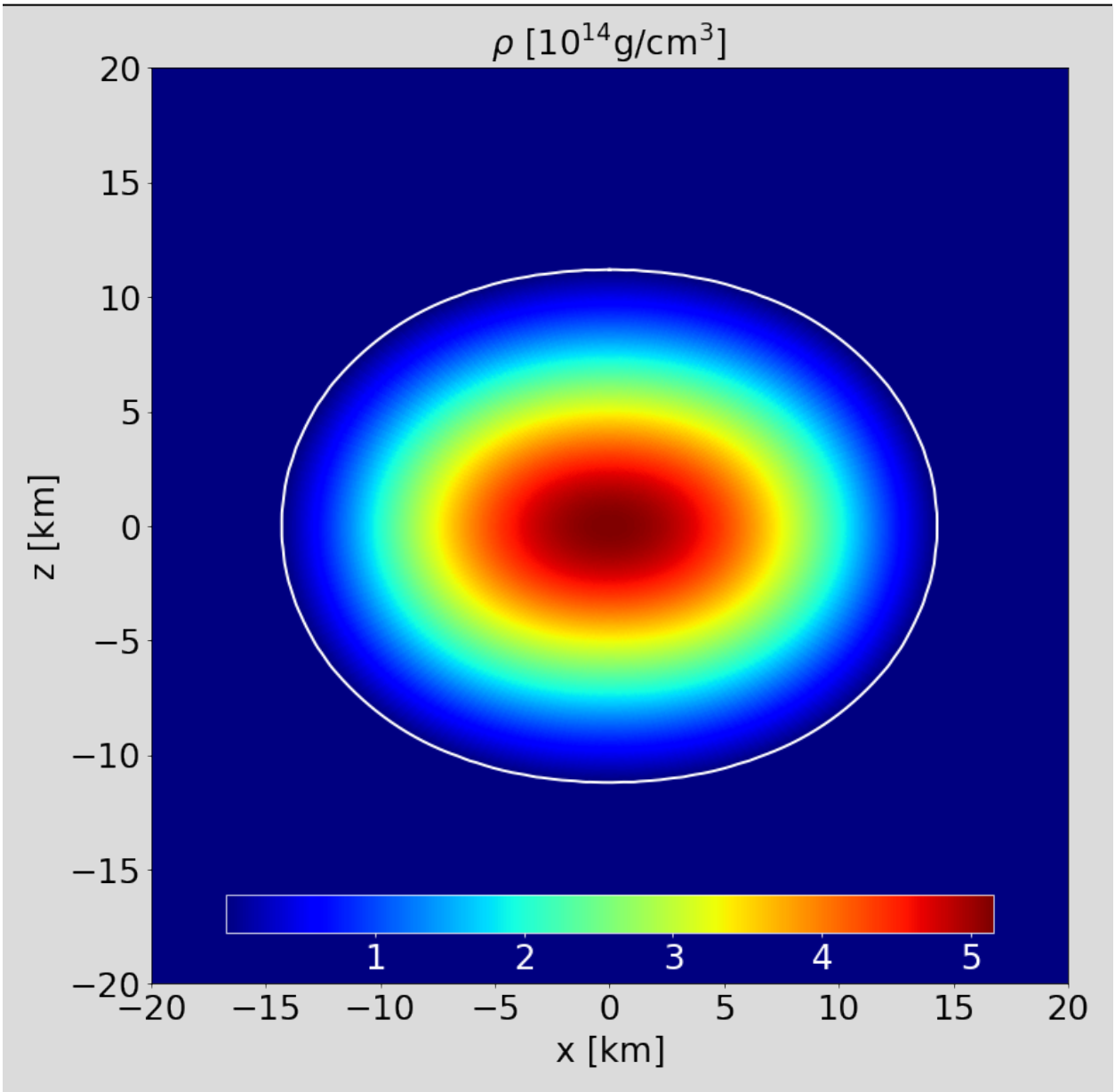
Here we present a few examples to show how to work with the code and the related performances. All cases have been run on a simple laptop

6.1 Non-rotating NS in GR with the POL2 EoS and a purely poloidal magnetic field

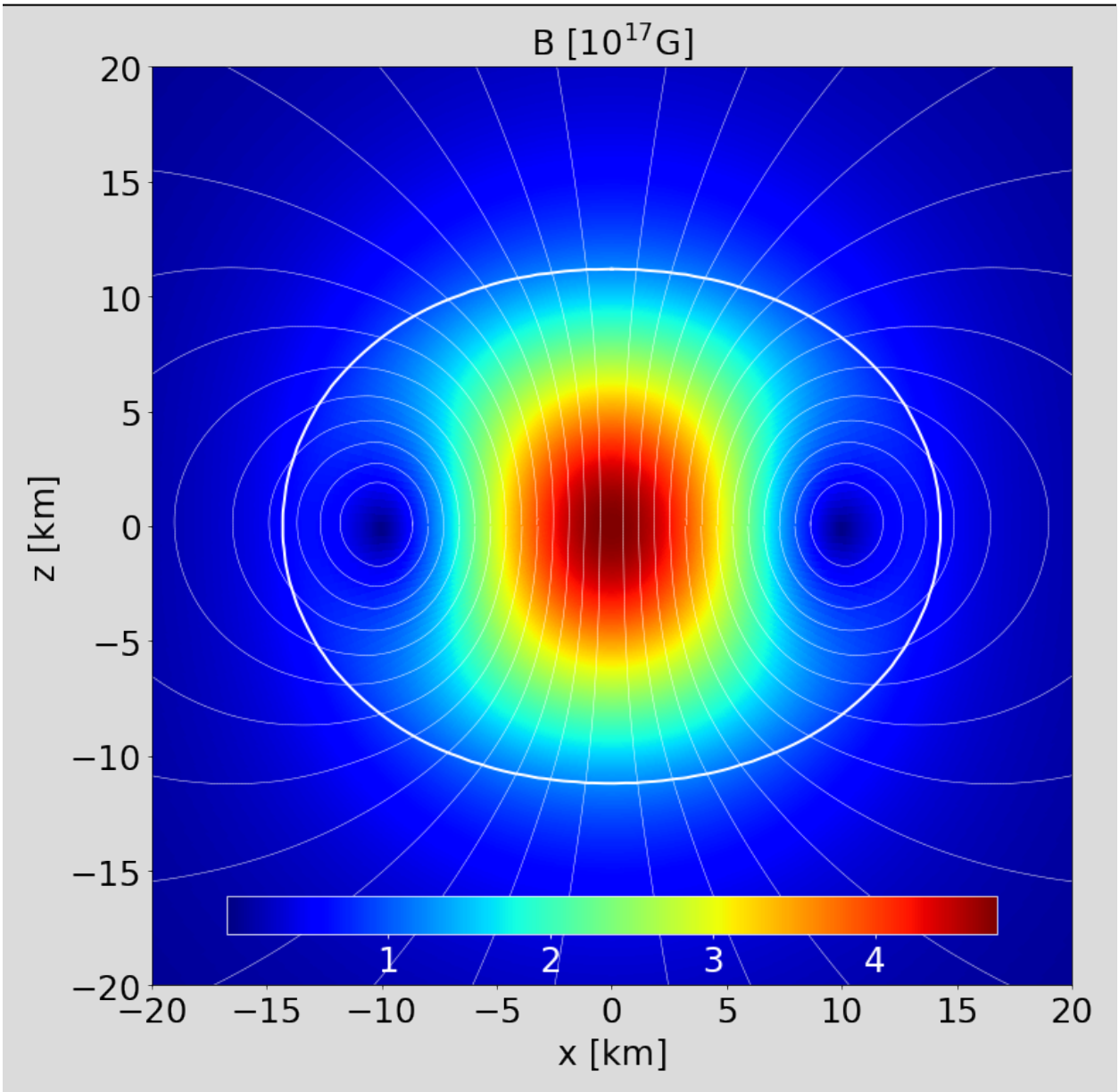
This is a model of an NS in GR described by the analytic POL2 EoS, endowed with a purely poloidal field. It has a J-frame central density $\rho_c = 8.3 \times 10^{-4}$ in code units (corresponding to 5.15×10^{14} gcm^{-3}) and a Komar mass in the E-frame of 1.442 M$_\odot$. The circumferential radius is 16.559 km.

Here are the main parameters used to compute this model. The detailed log with all the parameters can be found in the corresponding LogFile.dat.

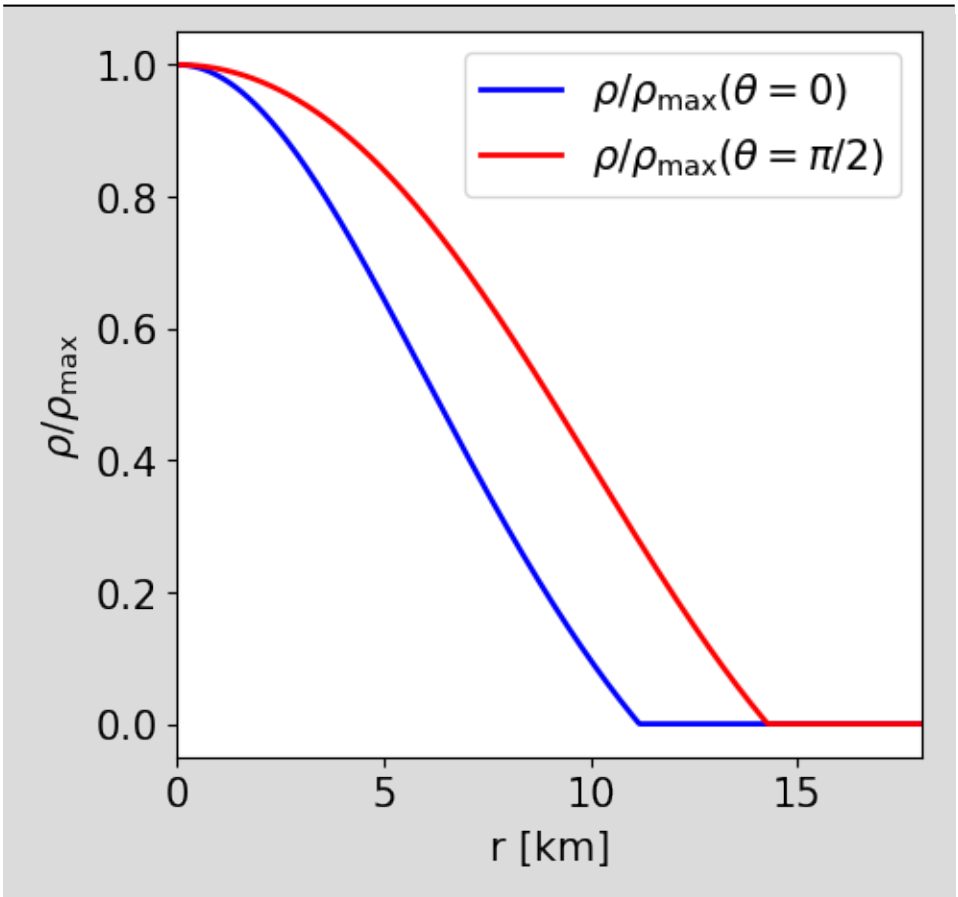
```
NR = 900, NTH = 100, NRREG = 600, MLS = 20, NGQ = 50, RREG = 13, RMAXSTR = 100,
RMAX = 100, REQMAX = 13.50, RHOINI = 8.30E-4, ALPHA0 = -2.0E-4, BETA0 = -6, GR = .FALSE.,
STRETCH = .TRUE., ANALYTIC = .TRUE., CONVHELP = .FALSE., QFACTOR = 0.85, QFACTORCHI = 0.
↪45,
QFACTORMETRIC = 0.35, QRELAX = 0.30, QAPHI = 0.50, EOSINT = .FALSE., K1 = 110.0, GAMMA =␣
↪2.0,
IMAG = .TRUE., IPOL = .TRUE., KBPOL = 0.34135, NPOL = 0.0, CSI = 0.0
```



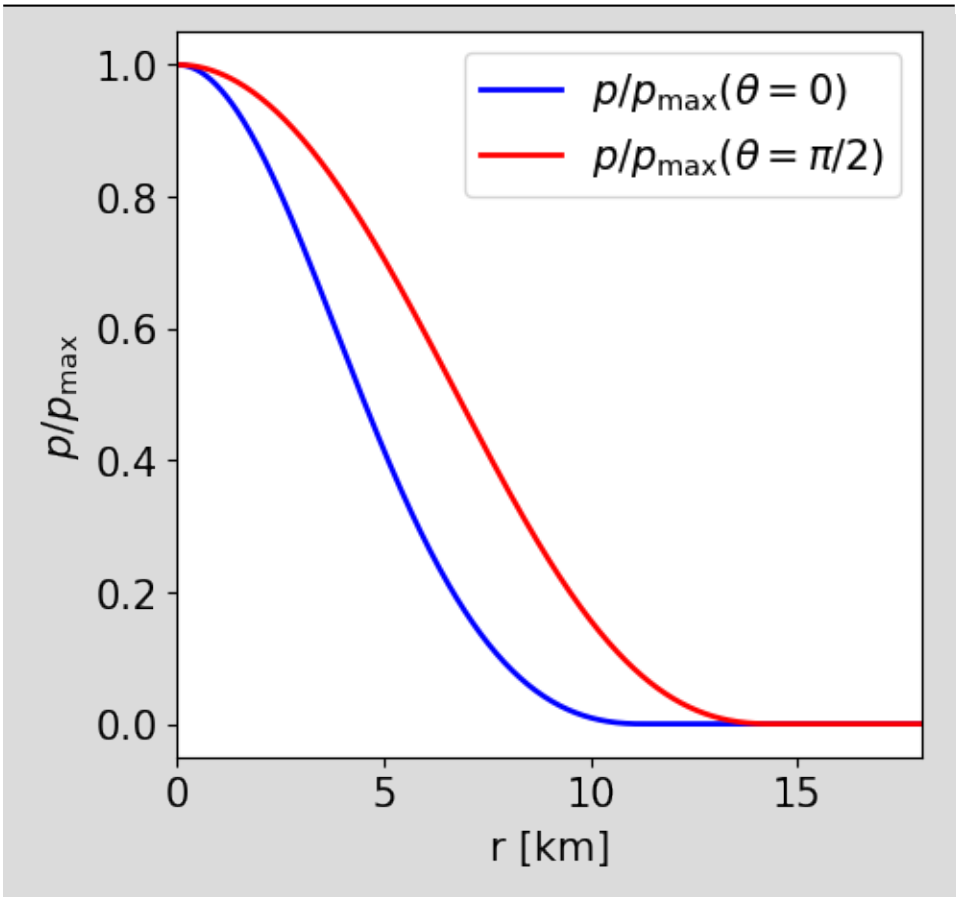
Section of the NS in the $x - z$ plane along with the contours the density.



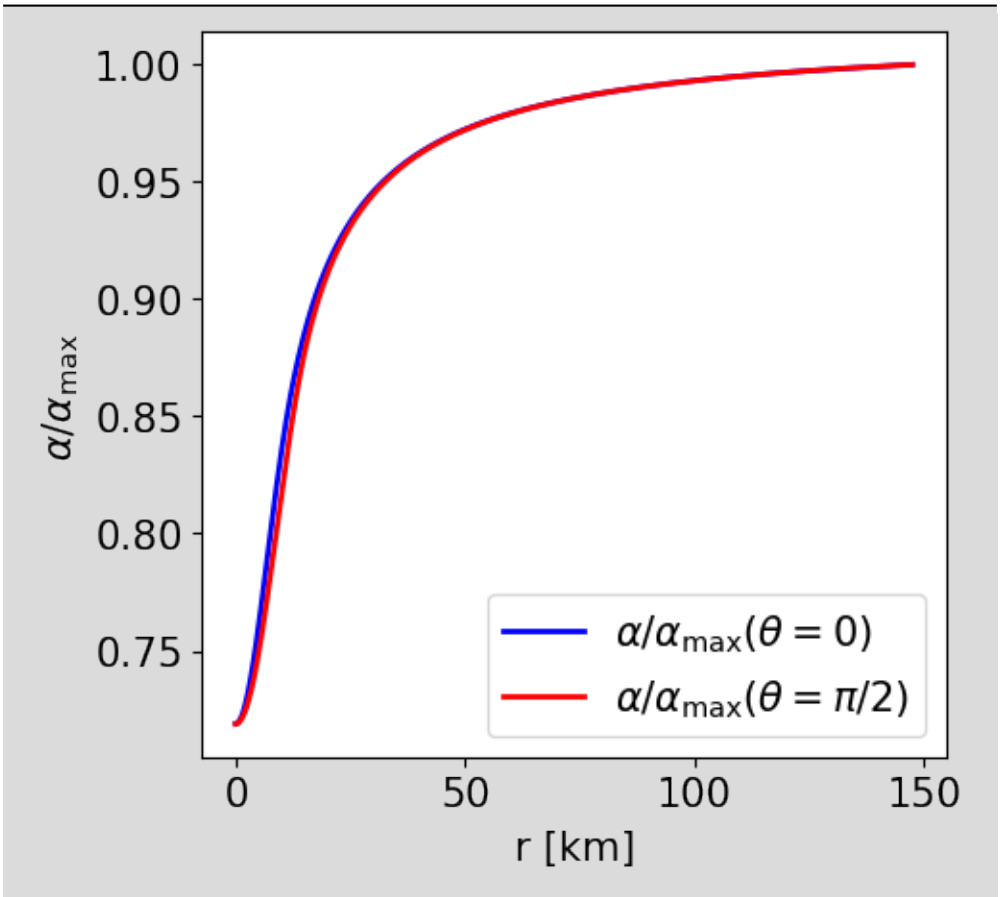
Section of the NS in the $x - z$ plane along with the contours the poloidal magnetic field strength and its field lines.



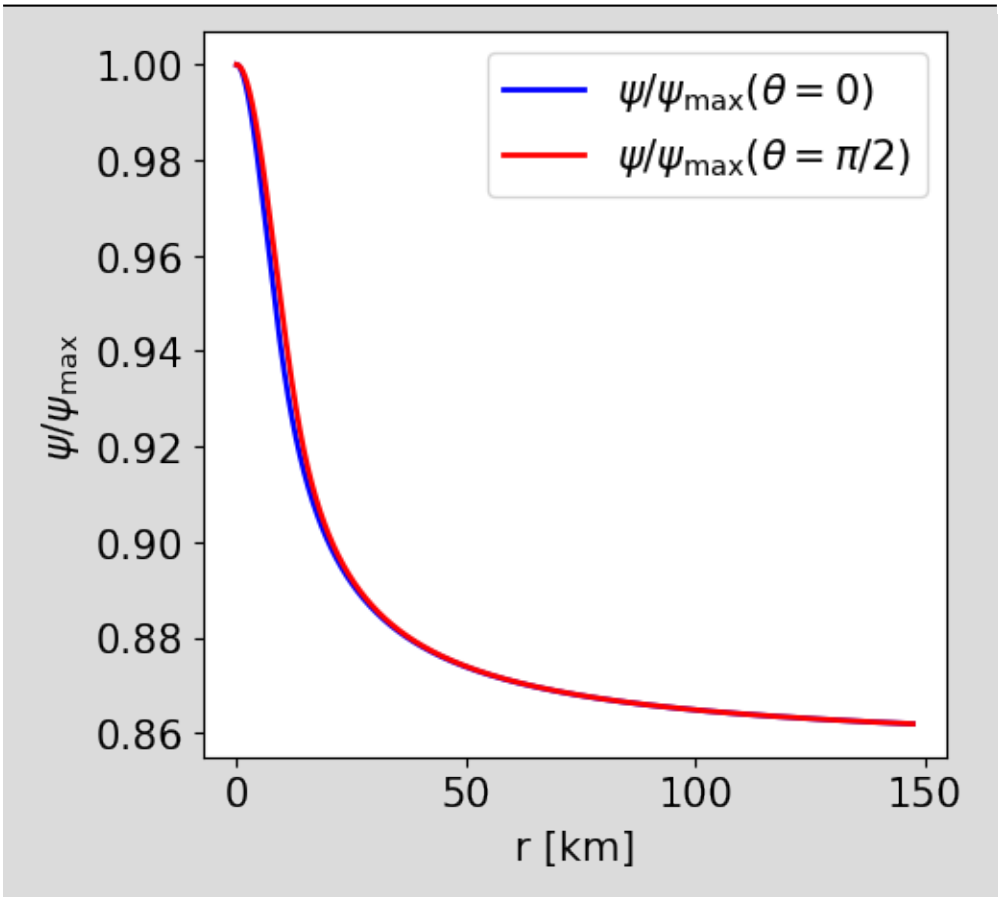
Radial profile of the NS density at the pole and at the equator.



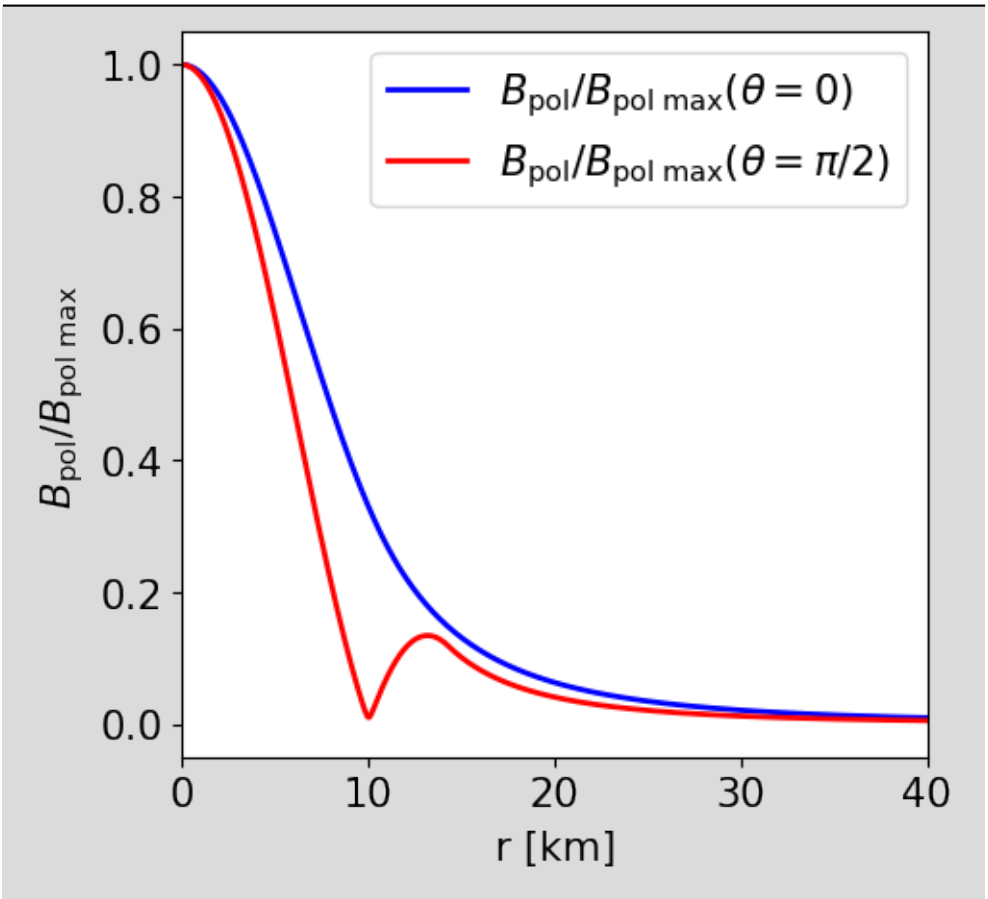
Radial profile of the NS pressure at the pole and at the equator.



Radial profile of the lapse function at the pole and at the equator.



Radial profile of the conformal factor at the pole and at the equator.



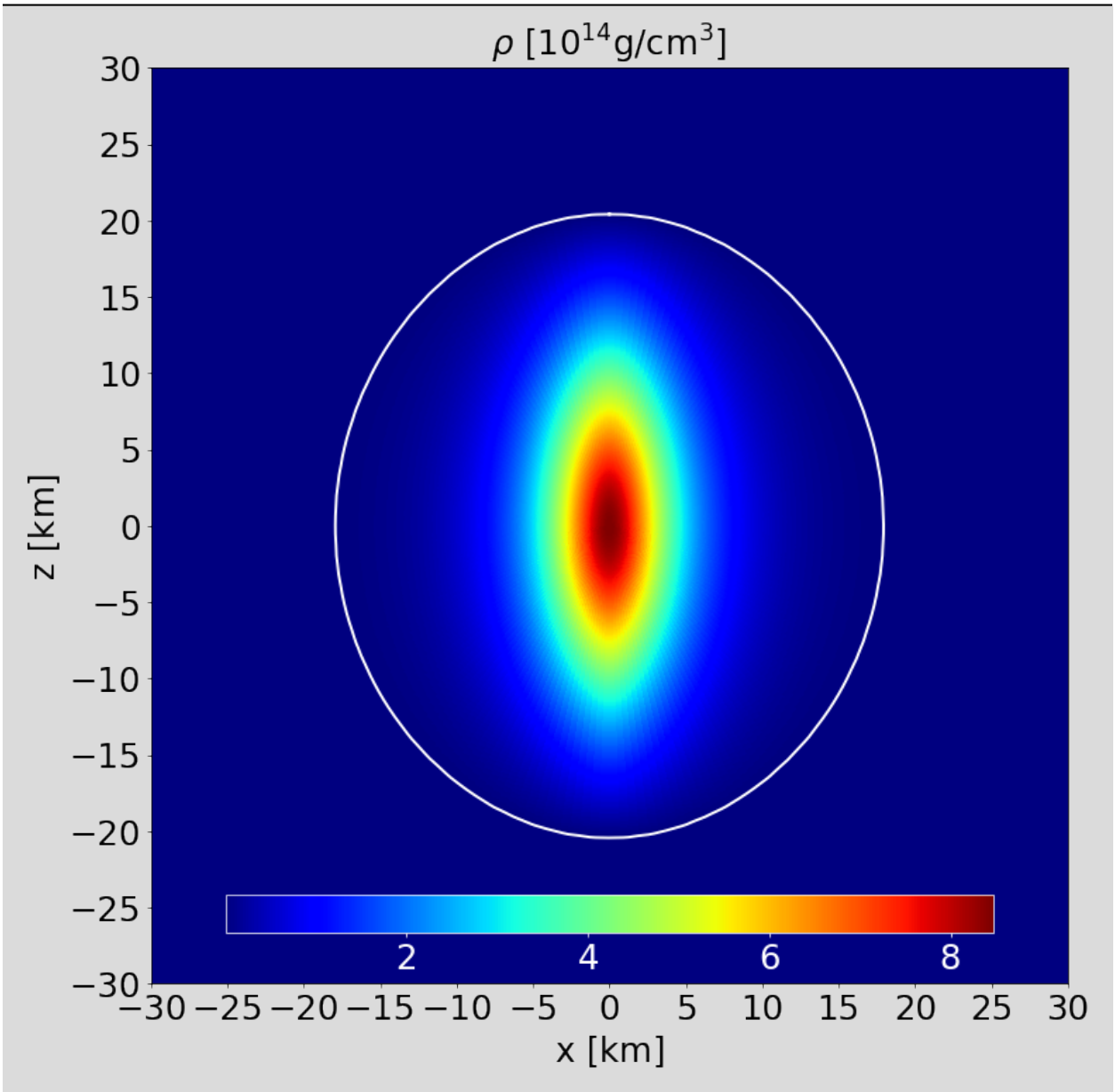
Radial profile of the poloidal magnetic field strength at the pole and at the equator.

6.2 Non-rotating NS in GR with the POL2 EoS and a purely toroidal magnetic field

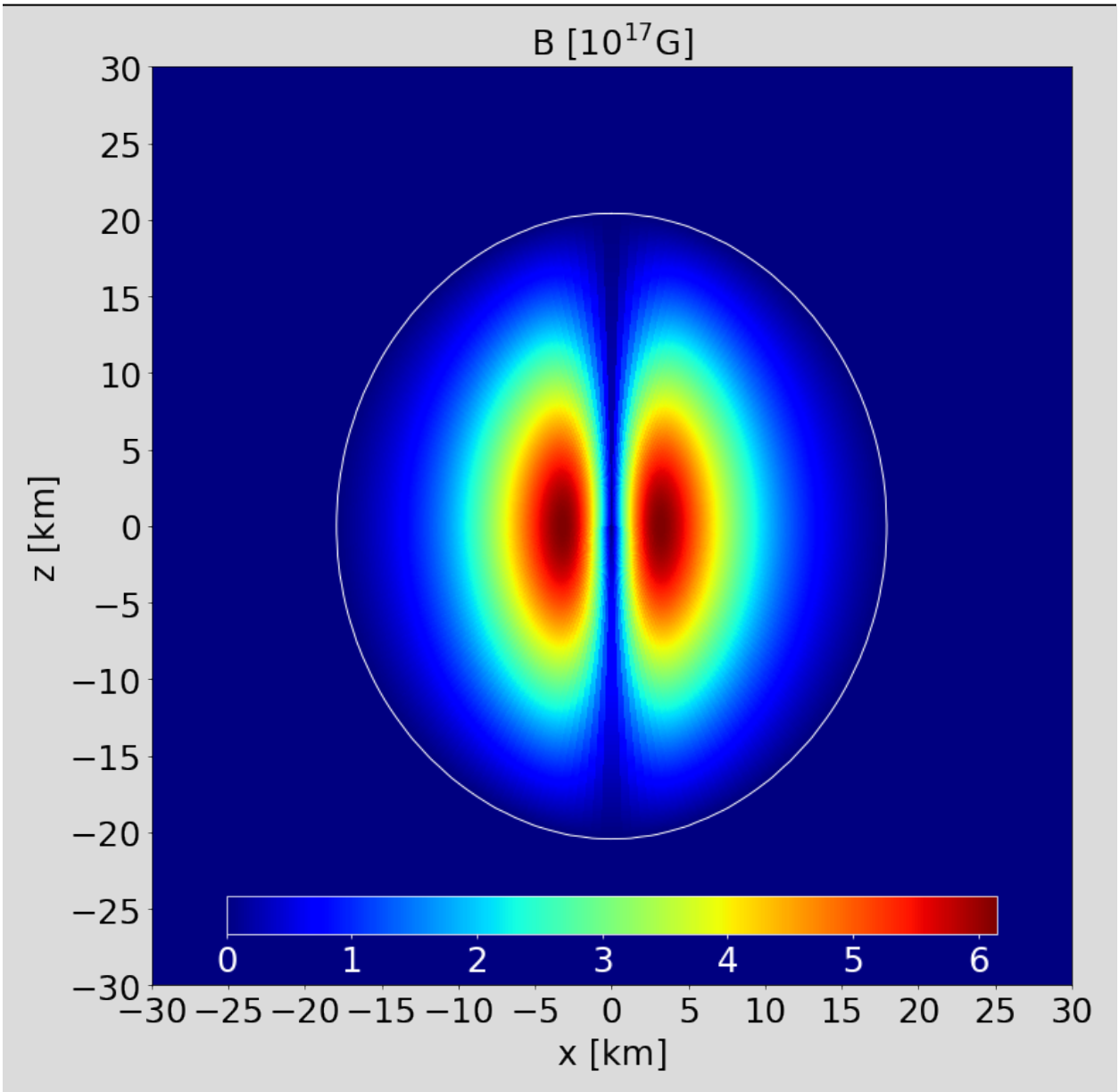
This is a model of an NS in GR described by the analytic POL2 EoS, endowed with a purely toroidal field. It has a central density $\rho_c = 1.366 \times 10^{-3}$ in code units (corresponding to 8.473×10^{14} gcm^{-3}) and a Komar mass of 1.597 M$_\odot$. The circumferential radius is 20.264 km.

Here are the main parameters used to compute this model. The detailed log with all the parameters can be found in the corresponding LogFile.dat.

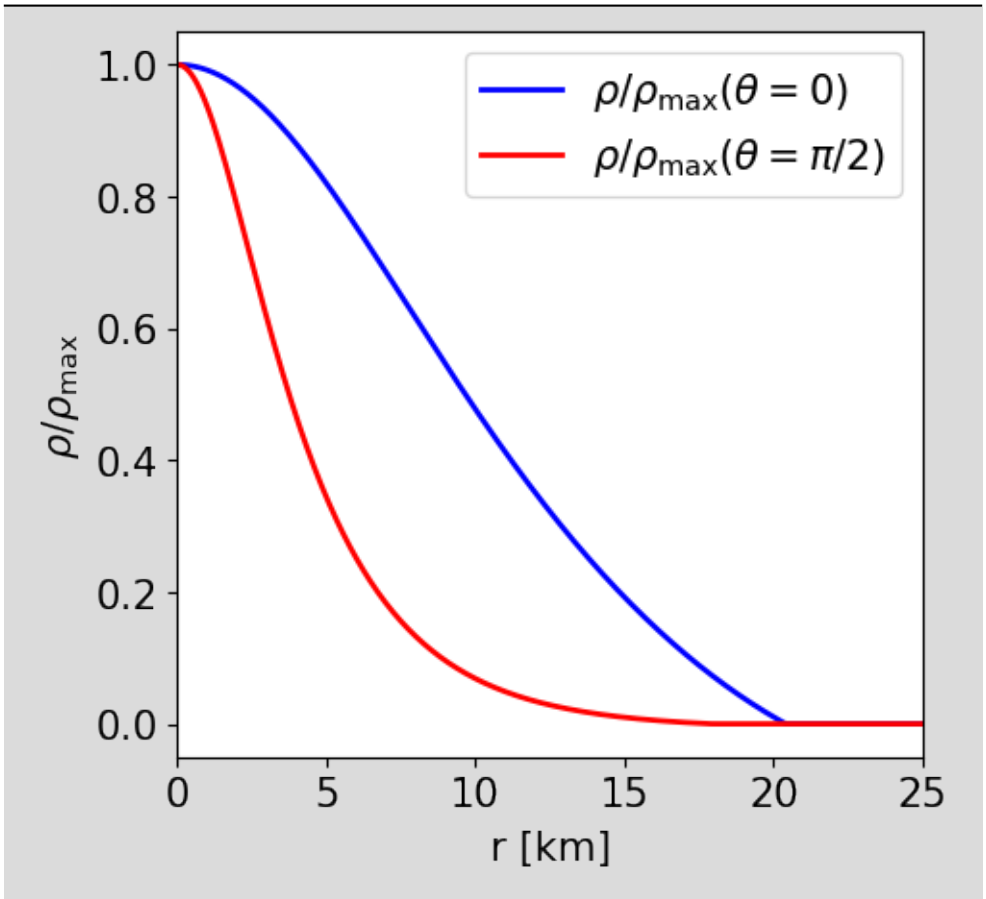
```
NR = 900, NTH = 100, NRREG = 600, MLS = 20, NGQ = 50, RREG = 13, RMAXSTR = 100,
RMAX = 100, REQMAX = 15.50, RHOINI = 1.366E-3, ALPHA0 = 0.0, BETA0 = 0.0, GR = .TRUE.,
STRETCH = .TRUE., ANALYTIC = .TRUE., CONVHELP = .FALSE., QFACTOR = 0.85, QFACTORCHI = 0.
↪45,
QFACTORMETRIC = 0.35, QRELAX = 0.30, QAPHI = 0.50, EOSINT = .FALSE., K1 = 110.0, GAMMA =␣
↪2.0,
IMAG = .TRUE., ITOR = .TRUE., BCOEF = 3.746, NPOL = 0.0, CSI = 0.0
```



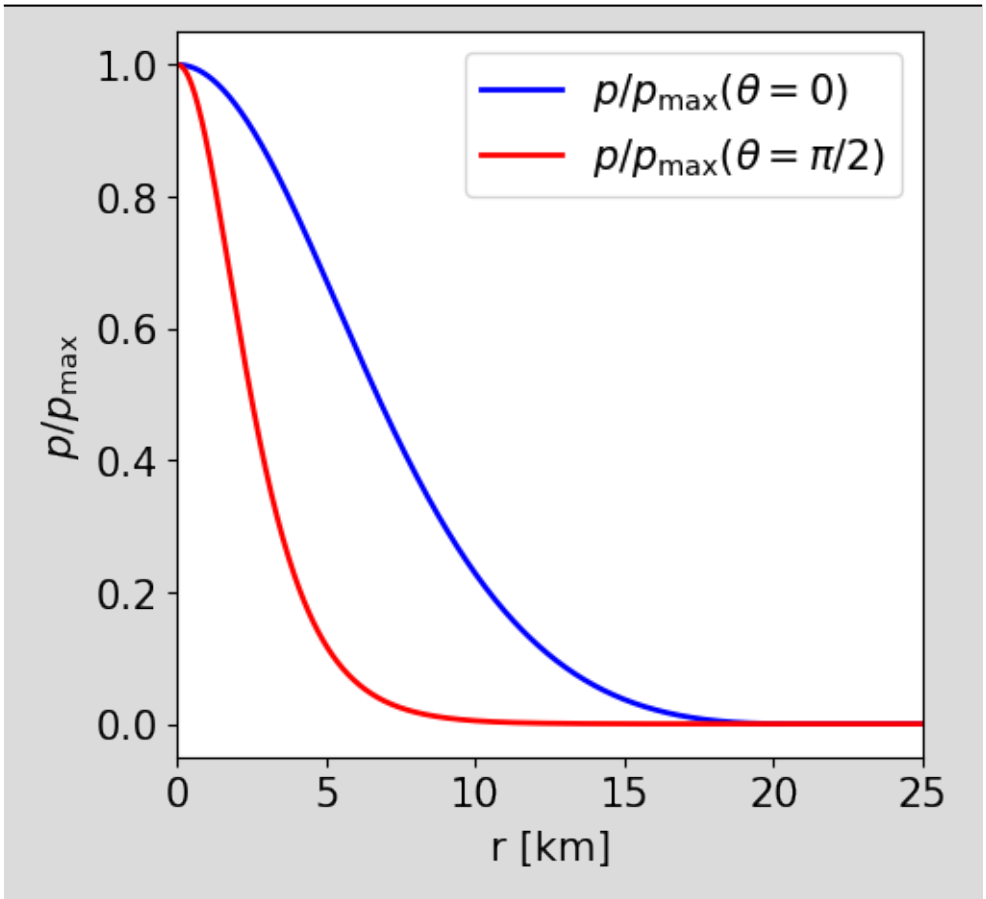
Section of the NS in the $x - z$ plane along with the contours the density.



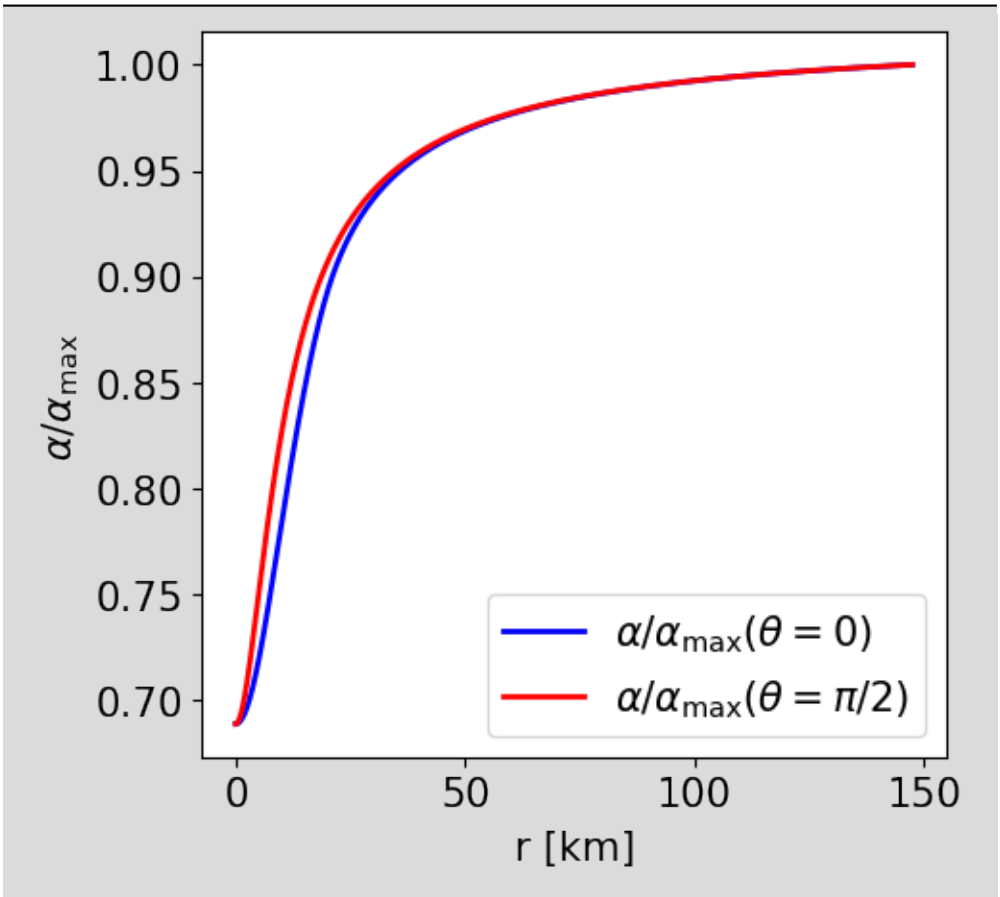
Section of the NS in the $x - z$ plane along with the contours the toroidal magnetic field strength.



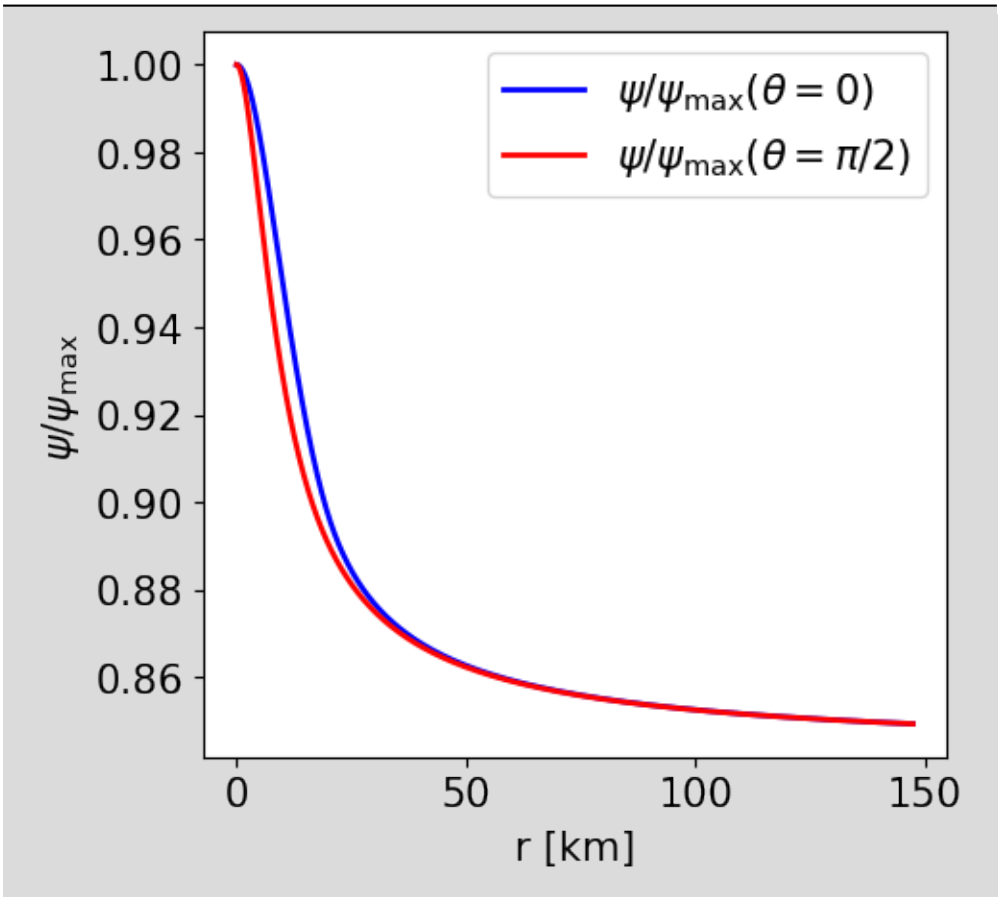
Radial profile of the NS density at the pole and at the equator.



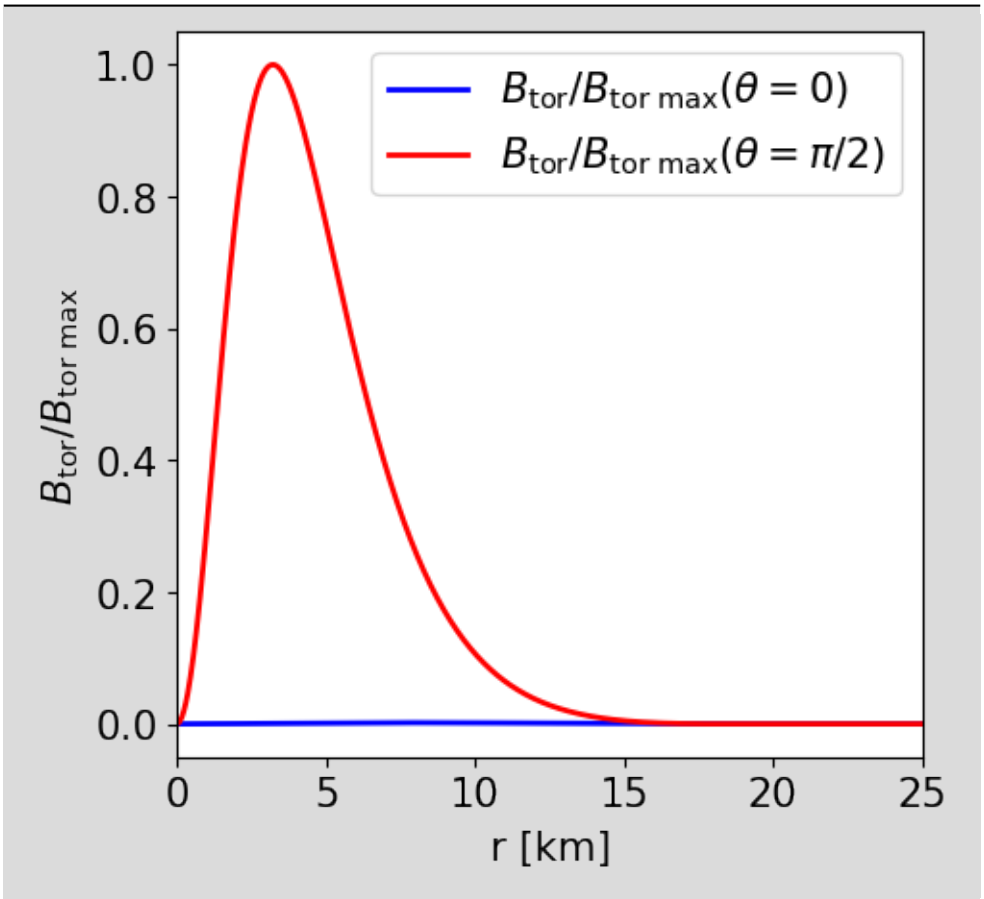
Radial profile of the NS pressure at the pole and at the equator.



Radial profile of the lapse function at the pole and at the equator.



Radial profile of the conformal factor at the pole and at the equator.



Radial profile of the toroidal magnetic field strength at the pole and at the equator.

6.3 Non-rotating NS in GR with the POL2 EoS and a twisted magnetosphere

This is a model of an NS in GR described by the analytic POL2 EoS, endowed with a purely toroidal field. It has a central density $\rho_c = 1.366 \times 10^{-3}$ in code units (corresponding to 8.473×10^{14} gcm^{-3}) and a Komar mass of 1.597 M$_\odot$. The circumferential radius is 20.264 km.

Here are the main parameters used to compute this model. The detailed log with all the parameters can be found in the corresponding LogFile.dat.

```
NR = 900, NTH = 100, NRREG = 600, MLS = 20, NGQ = 50, RREG = 13, RMAXSTR = 100,
RMAX = 100, REQMAX = 15.50, RHOINI = 1.366E-3, ALPHA0 = 0.0, BETA0 = 0.0, GR = .TRUE.,
STRETCH = .TRUE., ANALYTIC = .TRUE., CONVHELP = .FALSE., QFACTOR = 0.85, QFACTORCHI = 0.
↪45,
QFACTORMETRIC = 0.35, QRELAX = 0.30, QAPHI = 0.50, EOSINT = .FALSE., K1 = 110.0, GAMMA =␣
↪2.0,
IMAG = .TRUE., ITOR = .TRUE., BCOEF = 3.746, NPOL = 0.0, CSI = 0.0
```

6.4 Uniformly Rotating, unmagnetised NS in GR with the Pol2 EoS

Description.

Here are the main parameters used to compute this model. The detailed log with all the parameters can be found in the corresponding LogFile.dat.

```
PARAM1 = VALUE1, PARAM2 = VALUE2, PARAM3 = VALUE3, PARAM4 = VALUE4, ...
```

6.5 Differnetially Rotating, unmagnetised NS in GR with the APR EoS

```
PARAM1 = VALUE1, PARAM2 = VALUE2, PARAM3 = VALUE3, PARAM4 = VALUE4, ...
```

6.6 Differentially Rotating, toroidal magnetised NS in GR with the Pol2 EoS

Description.

Here are the main parameters used to compute this model. The detailed log with all the parameters can be found in the corresponding LogFile.dat.

```
PARAM1 = VALUE1, PARAM2 = VALUE2, PARAM3 = VALUE3, PARAM4 = VALUE4, ...
```

6.7 Uniformly Rotating, poloidal magnetised NS in GR with the Pol2 EoS

Description.

Here are the main parameters used to compute this model. The detailed log with all the parameters can be found in the corresponding LogFile.dat.

```
PARAM1 = VALUE1, PARAM2 = VALUE2, PARAM3 = VALUE3, PARAM4 = VALUE4, ...
```

CHAPTER

SEVEN

EXAMPLES IN STTS

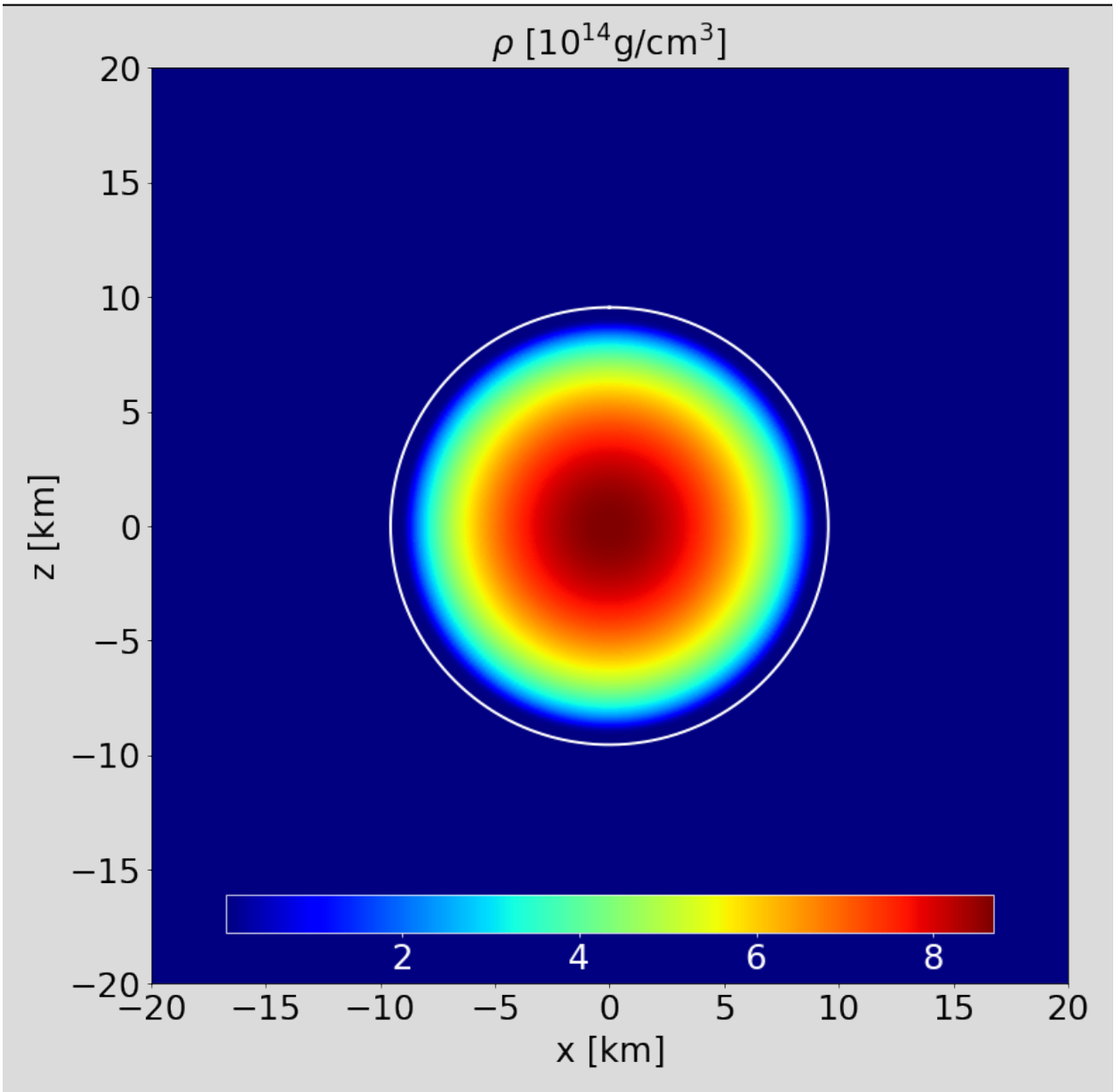
Here we present a few examples to show how to work with the code and the related performances. All cases have been run on a laptop.

7.1 Non-rotating, unmagnetised NS in STT with the APR EoS

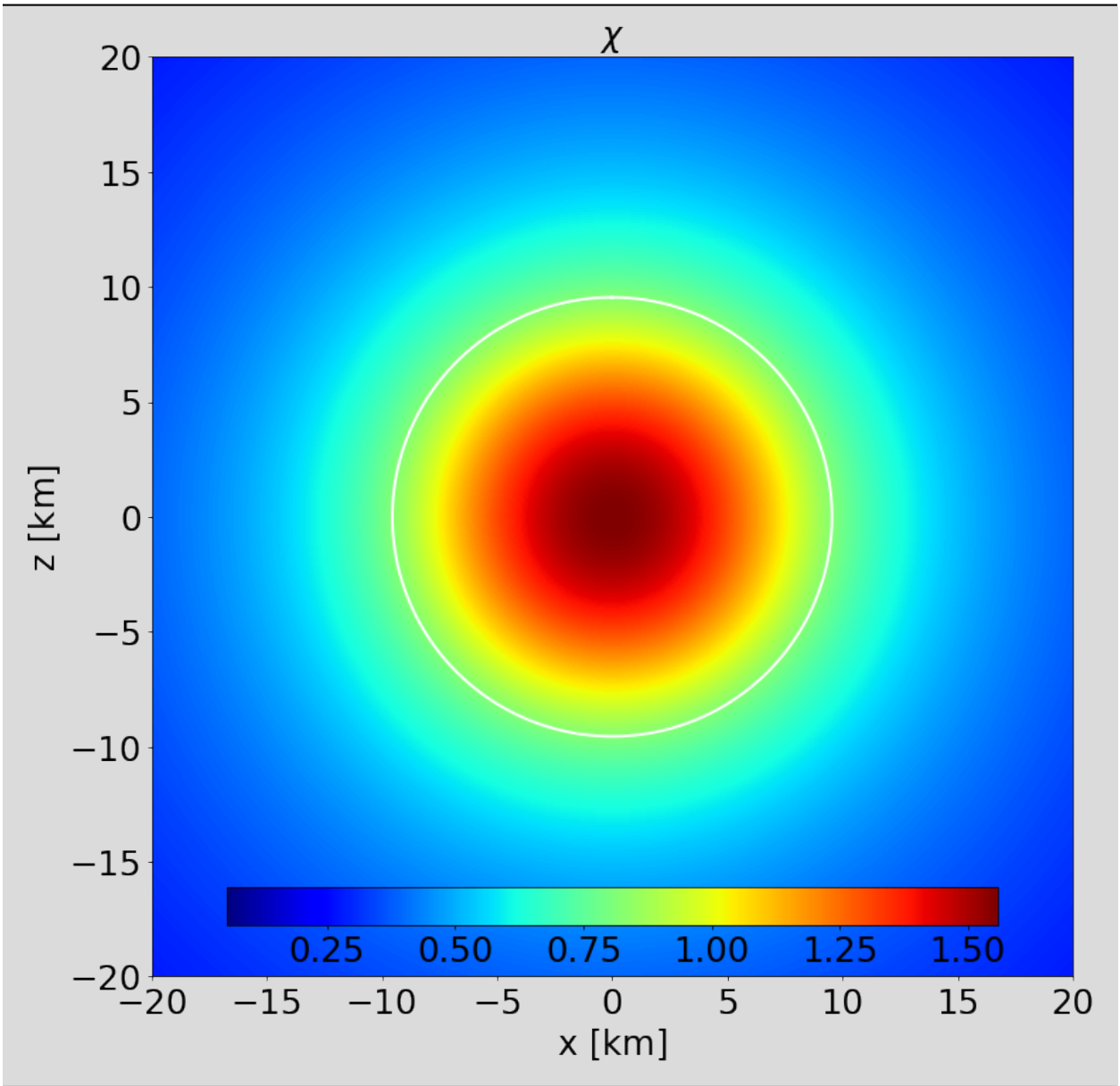
This is a model of an unmagnetised NS in STTs with $\alpha_0 = -2.0 \times 10^{-4}$ and $\beta_0 = -6$ described by the APR EoS. It has a J-frame central density $\rho_c = 1.4 \times 10^{-3}$ in code units (corresponding to 8.684×10^{14} gcm^{-3}) and a Komar mass in the E-frame of 1.086 M$_\odot$. The circumferential radius in the J-frame is 10.963 km, and the E-frame scalar charge is 0.532 M$_\odot$.

Here are the main parameters used to compute this model. The detailed log with all the parameters can be found in the corresponding LogFile.dat.

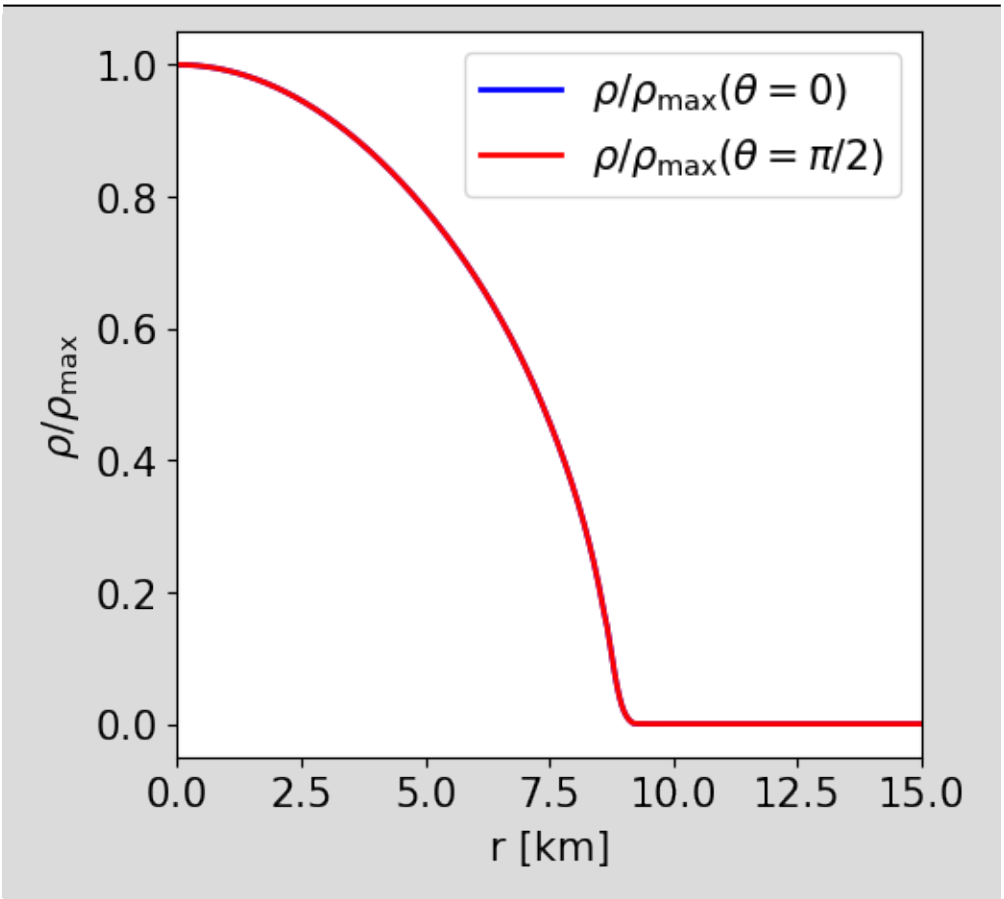
```
NR = 900, NTH = 100, NRREG = 600, MLS = 20, NGQ = 50, RREG = 10, RMAXSTR = 100,
RMAX = 100, REQMAX = 11.50, RHOINI = 1.40E-3, ALPHA0 = -2.0E-4, BETA0 = -6, GR = .FALSE.,
STRETCH = .TRUE., ANALYTIC = .TRUE., CONVHELP = .FALSE., QFACTOR = 0.85, QFACTORCHI = 0.
↪45,
QFACTORMETRIC = 0.35, QRELAX = 0.30, QAPHI = 0.50, EOSINT = .TRUE., FILEEOS = 'APR_
↪resampled.dat'
```



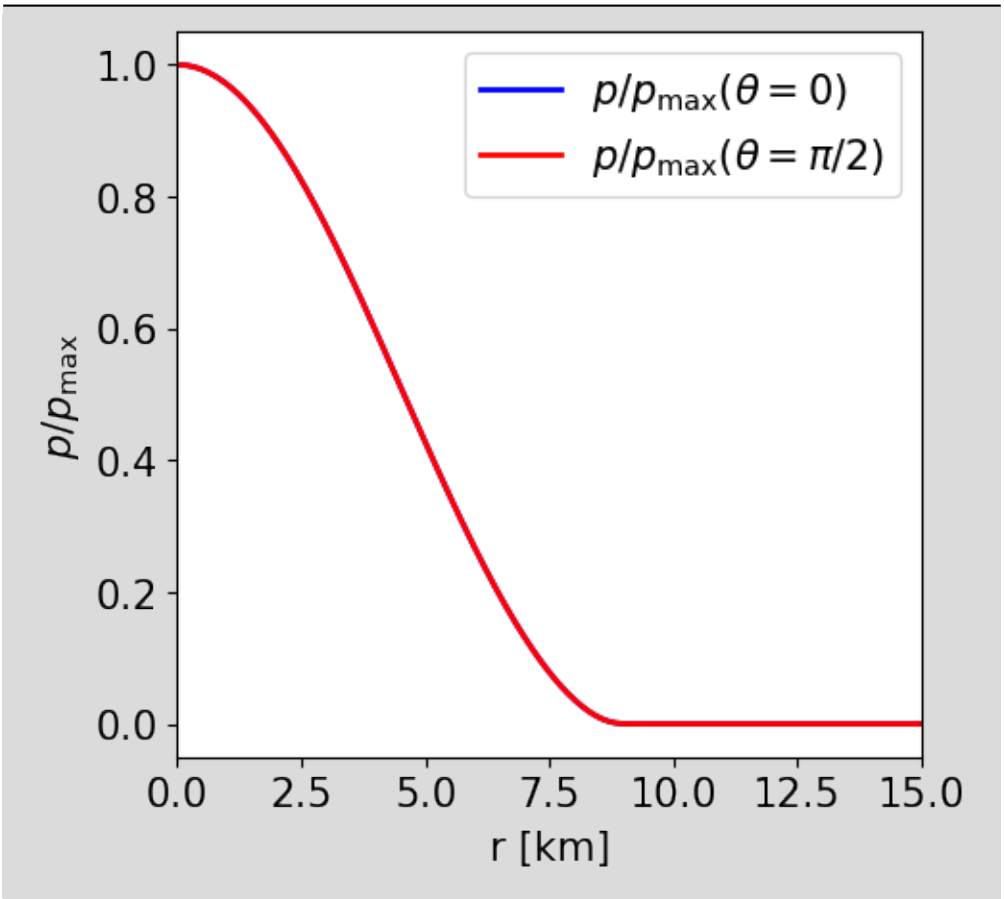
Section of the NS in the $x - z$ plane along with the contours the density.



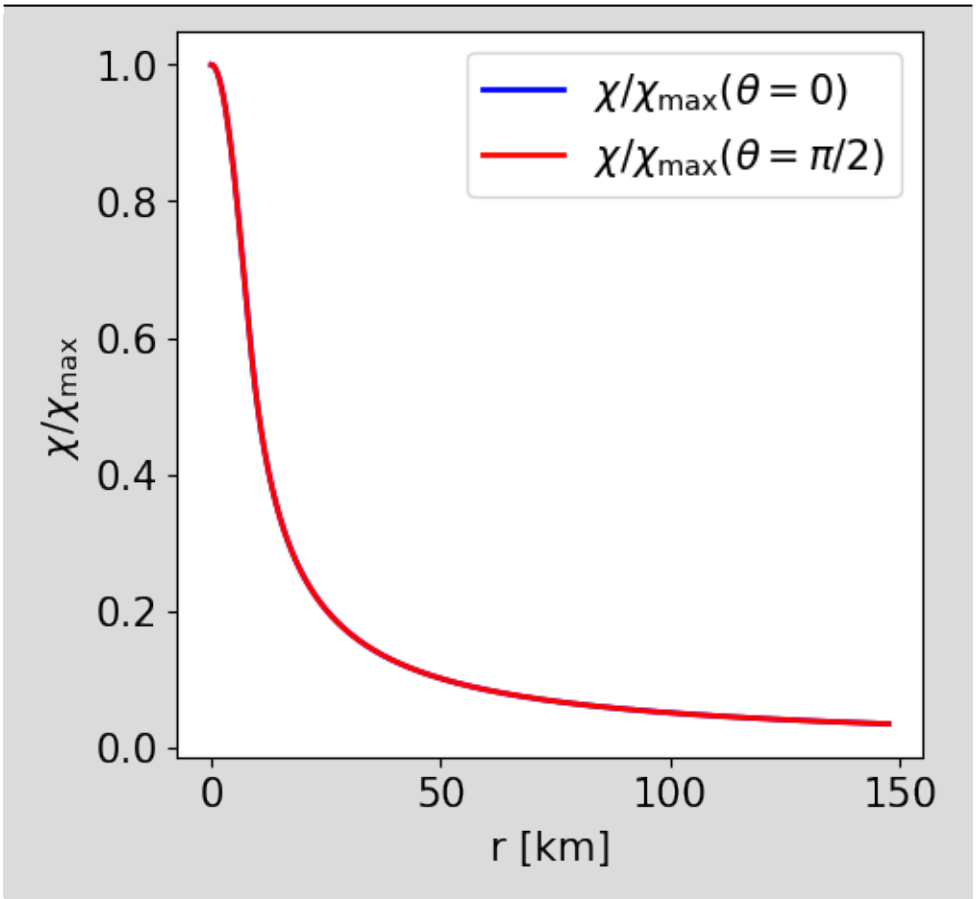
Section of the NS in the $x - z$ plane along with the contours the scalar field.



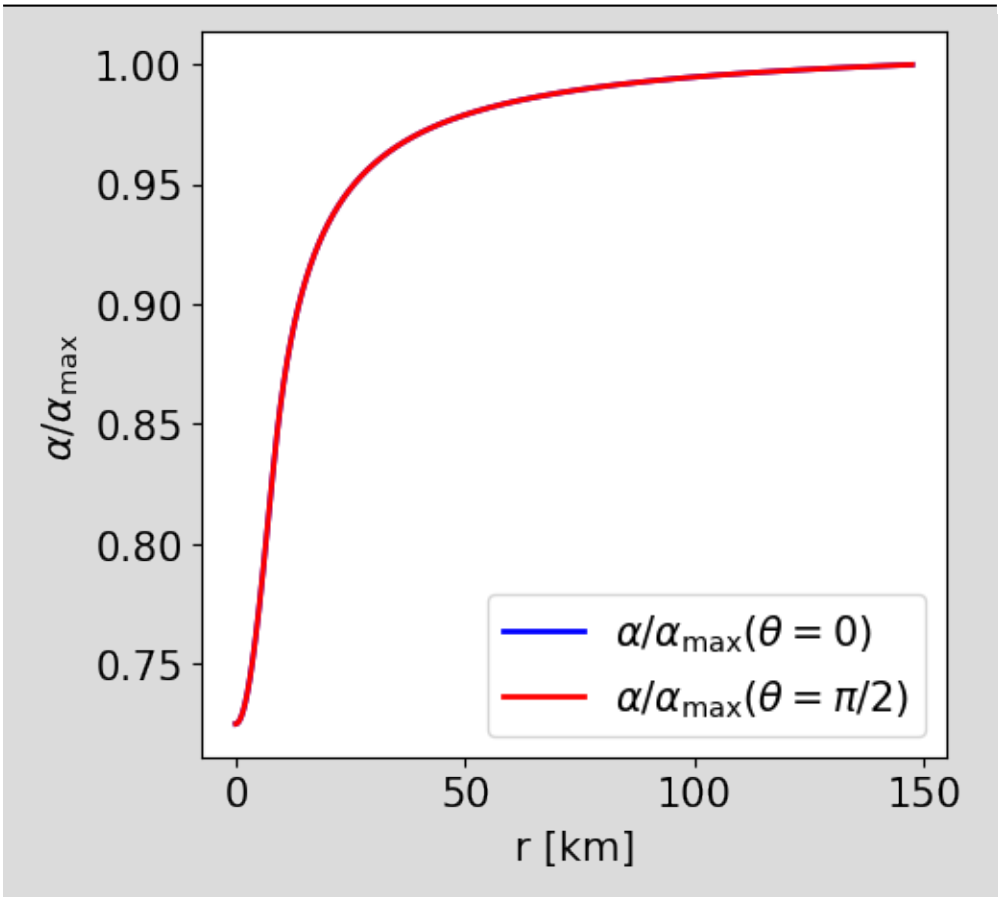
Radial profile of the NS density at the pole and at the equator.



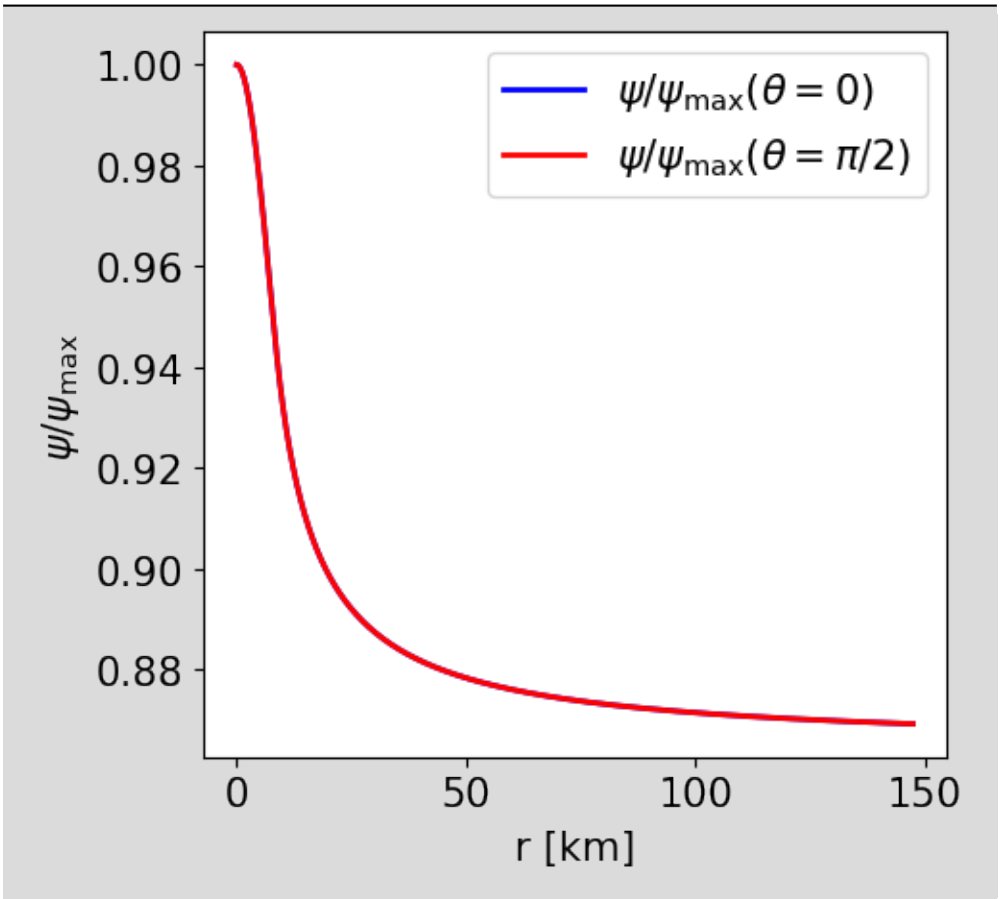
Radial profile of the NS pressure at the pole and at the equator.



Radial profile of the scalar field at the pole and at the equator.



Radial profile of the lapse function at the pole and at the equator.



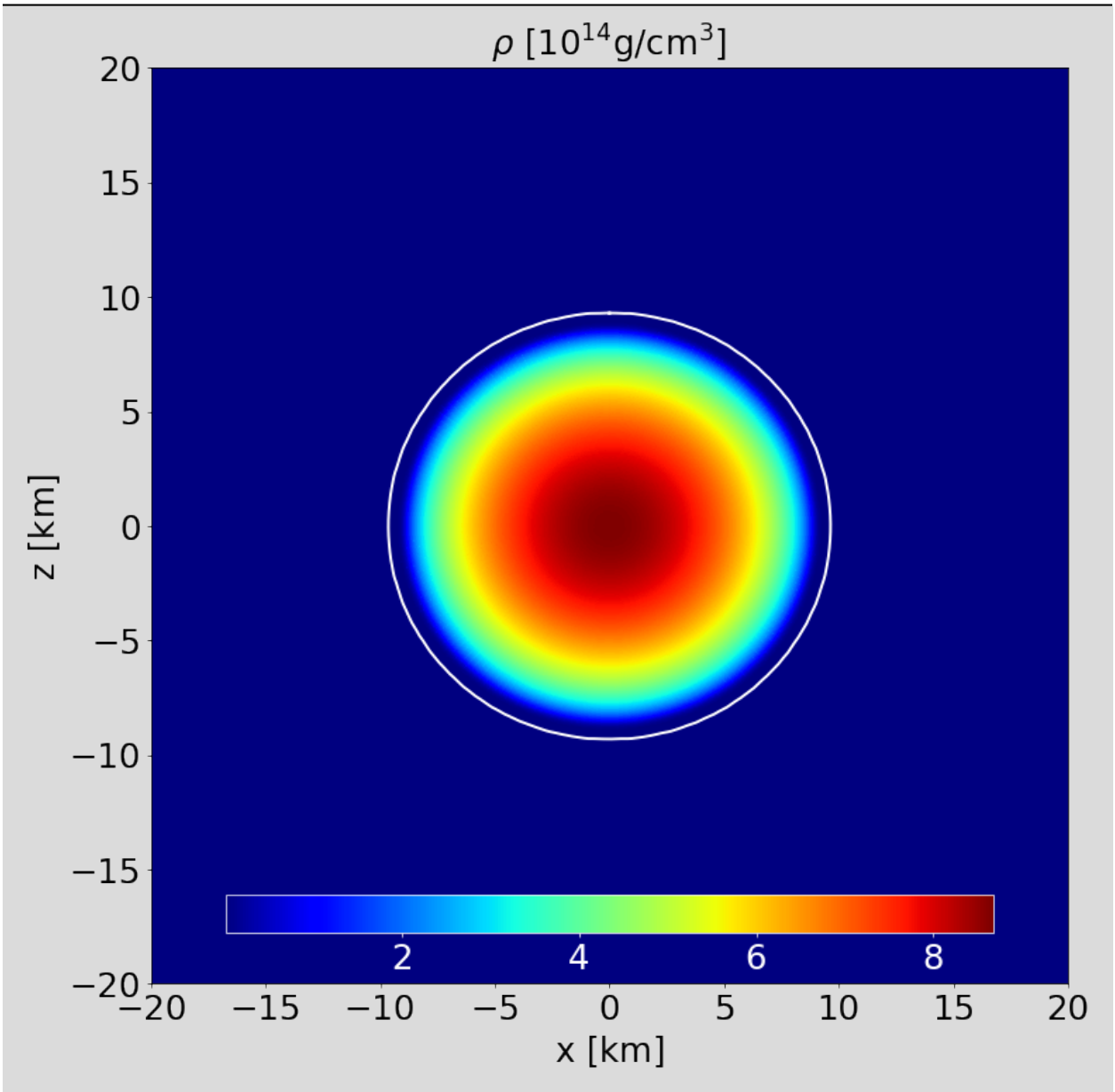
Radial profile of the conformal factor at the pole and at the equator.

7.2 Non-rotating NS in STT with the APR EoS and a purely poloidal magnetic field

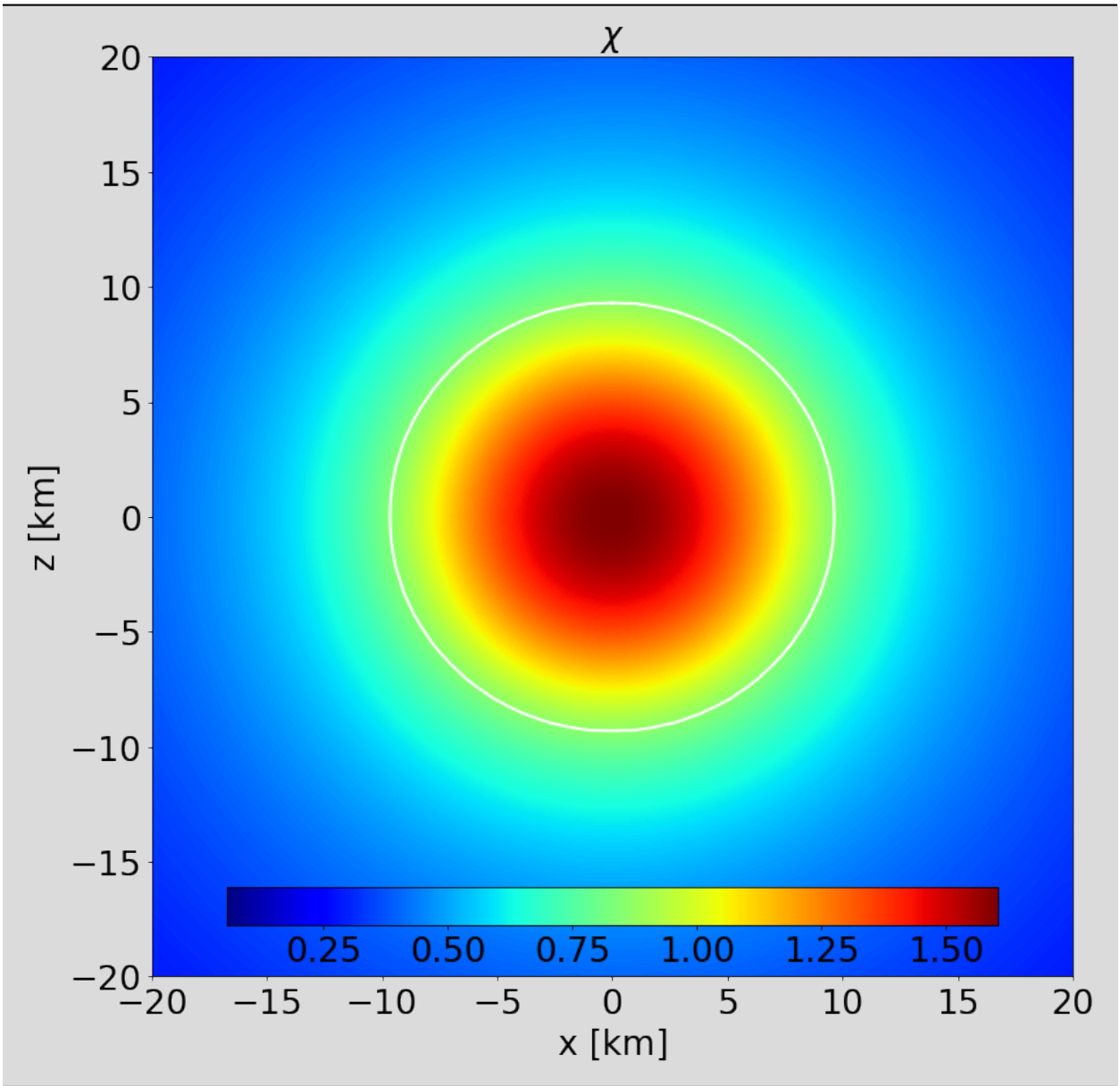
This is a model of an NS in STTs with $\alpha_0 = -2.0 \times 10^{-4}$ and $\beta_0 = -6$ described by the APR EoS, endowed with a purely poloidal field. It has a J-frame central density $\rho_c = 1.4 \times 10^{-3}$ in code units (corresponding to 8.684×10^{14} gcm^{-3}) and a Komar mass in the E-frame of 1.098 M$_\odot$. The circumferential radius in the J-frame is 11.072 km, and the E-frame scalar charge is 0.551 M$_\odot$.

Here are the main parameters used to compute this model. The detailed log with all the parameters can be found in the corresponding LogFile.dat.

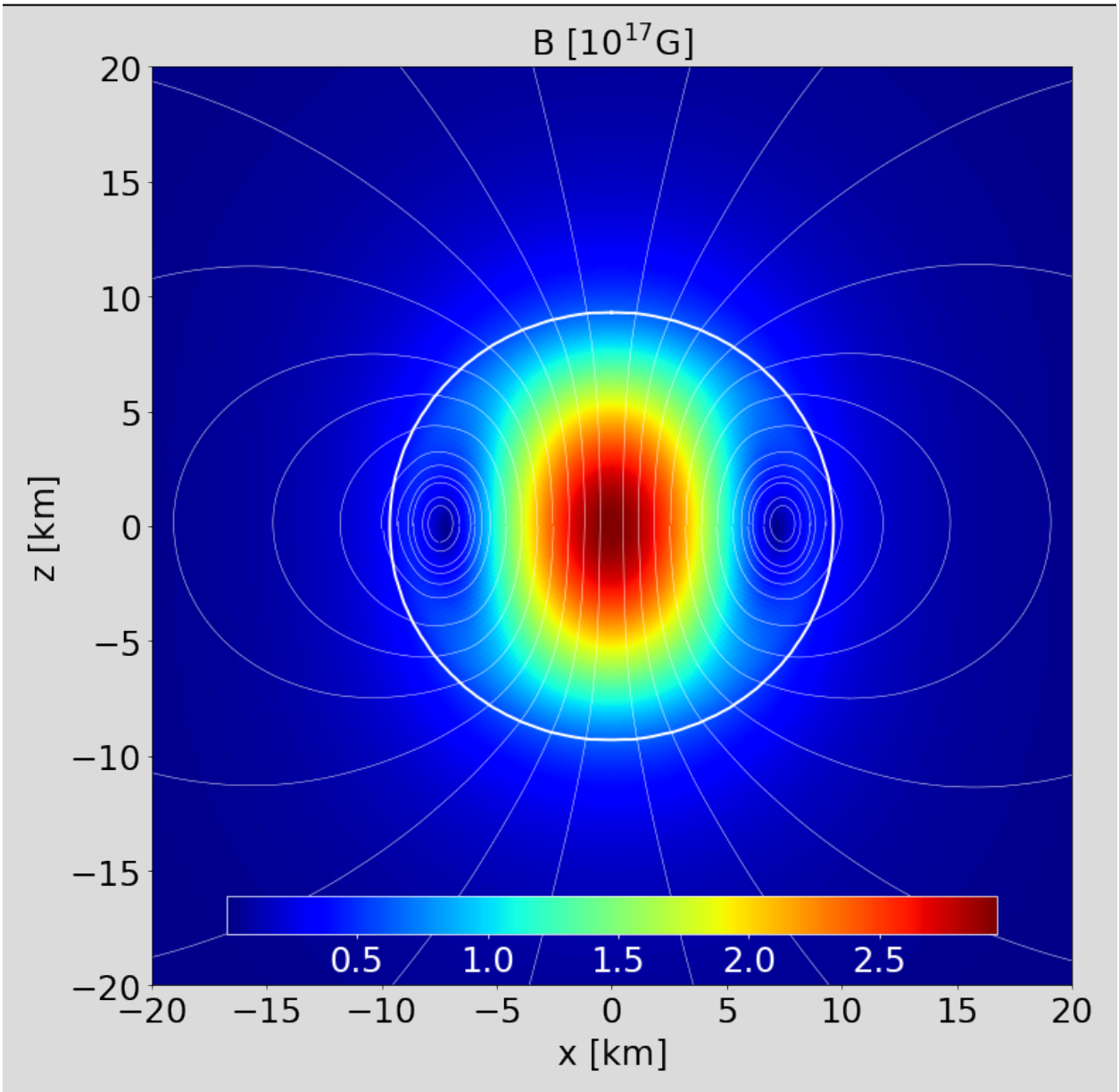
```
NR = 900, NTH = 100, NRREG = 600, MLS = 20, NGQ = 50, RREG = 10, RMAXSTR = 100,
RMAX = 100, REQMAX = 11.50, RHOINI = 1.40E-3, ALPHA0 = -2.0E-4, BETA0 = -6, GR = .FALSE.,
STRETCH = .TRUE., ANALYTIC = .TRUE., CONVHELP = .FALSE., QFACTOR = 0.85, QFACTORCHI = 0.
↪45,
QFACTORMETRIC = 0.35, QRELAX = 0.30, QAPHI = 0.50, EOSINT = .TRUE., FILEEOS = 'APR_
↪resampled.dat',
IMAG = .TRUE., IPOL = .TRUE., KBPOL = 0.2, NPOL = 0.0, CSI = 0.0
```



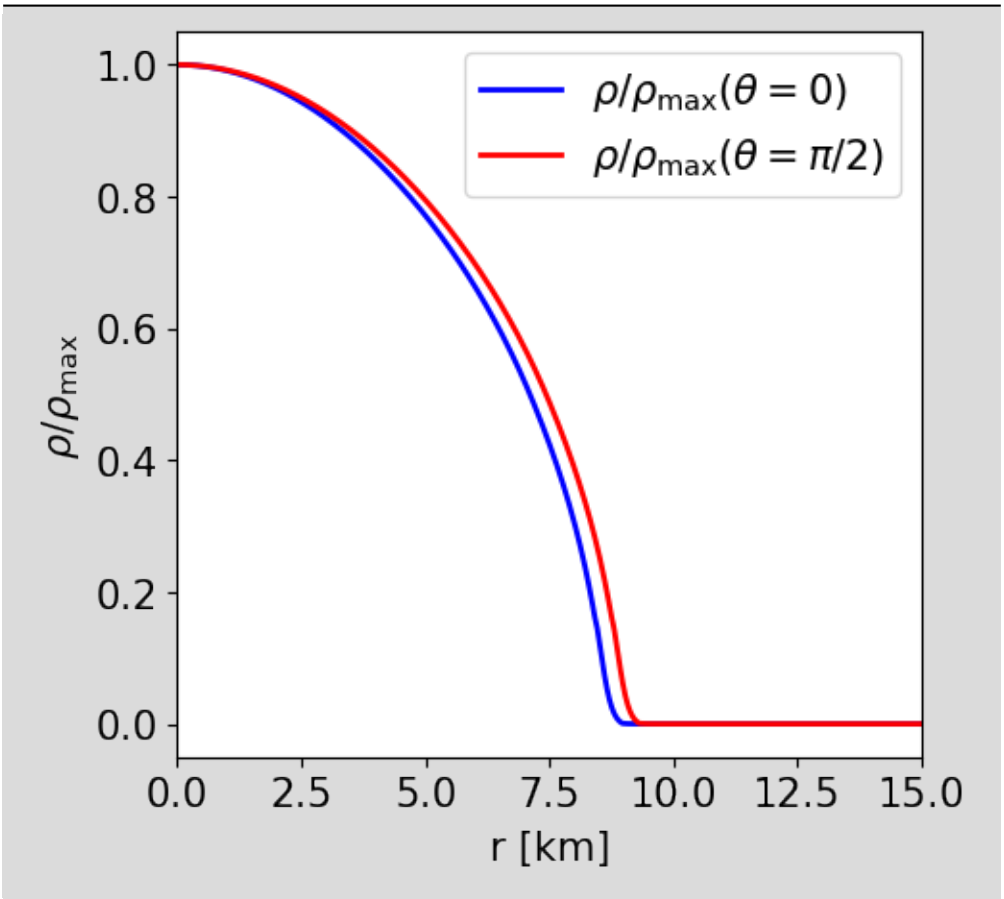
Section of the NS in the $x - z$ plane along with the contours the density.



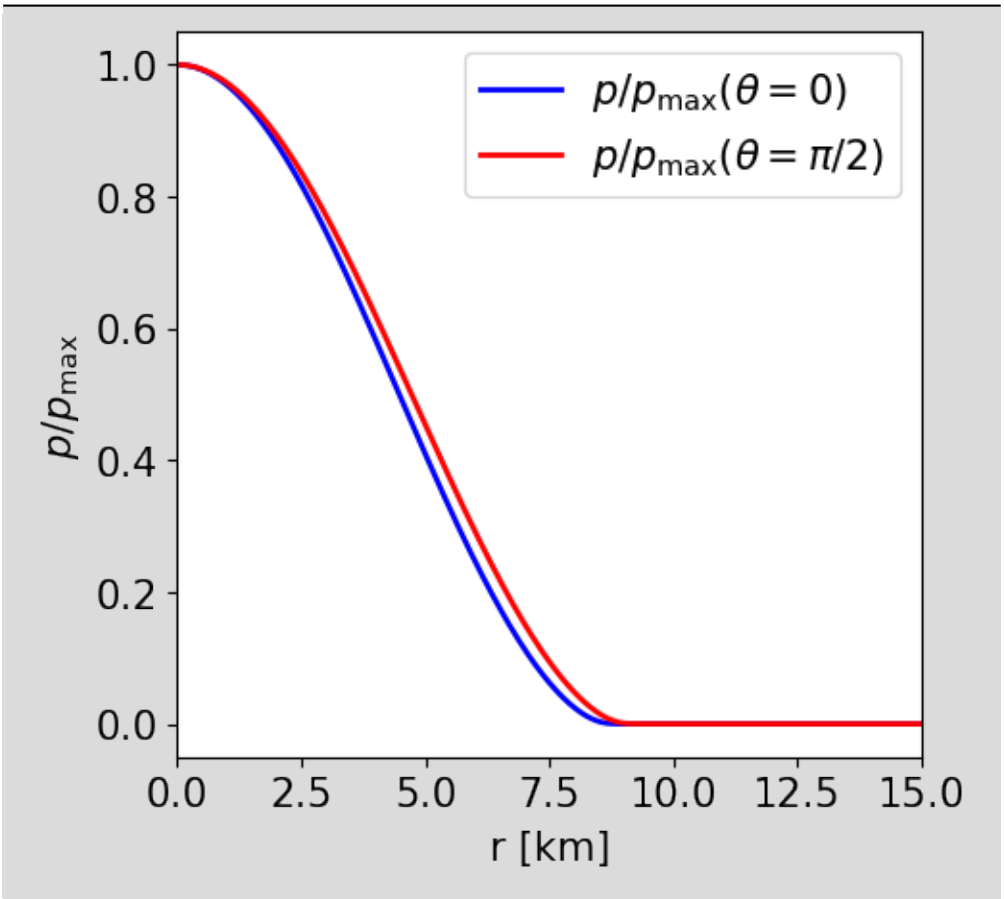
Section of the NS in the $x - z$ plane along with the contours the scalar field.



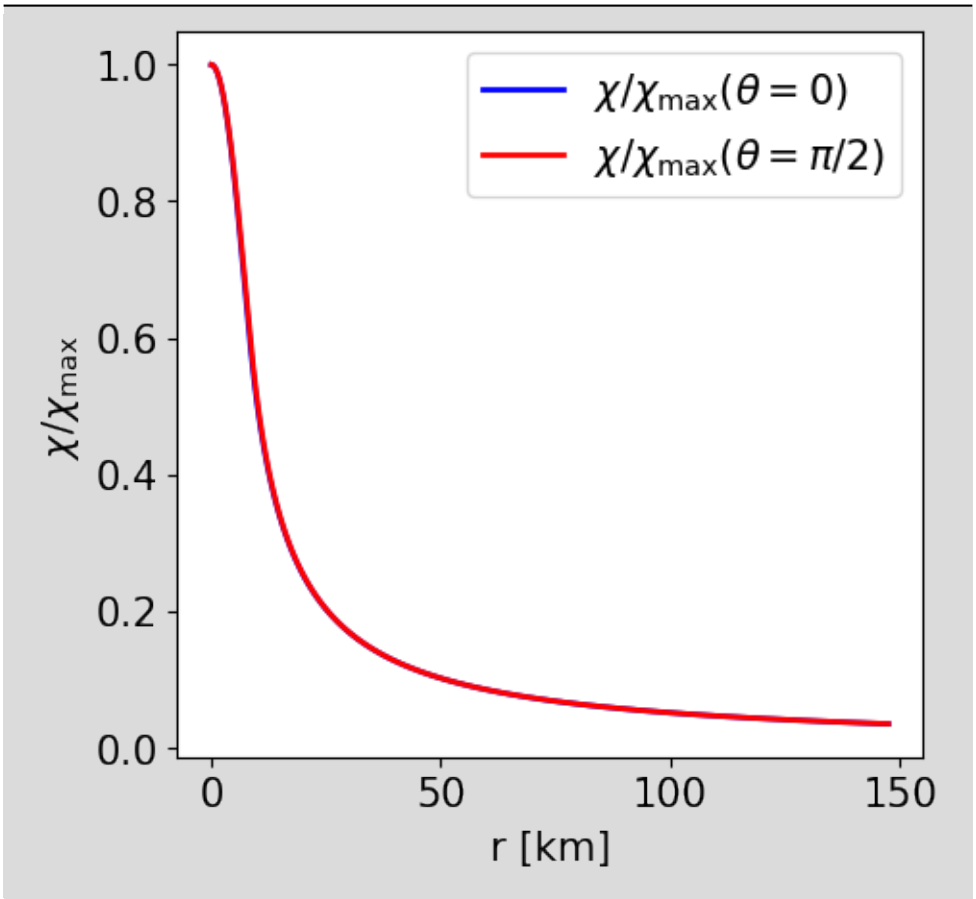
Section of the NS in the $x - z$ plane along with the contours the poloidal magnetic field strength and its field lines.



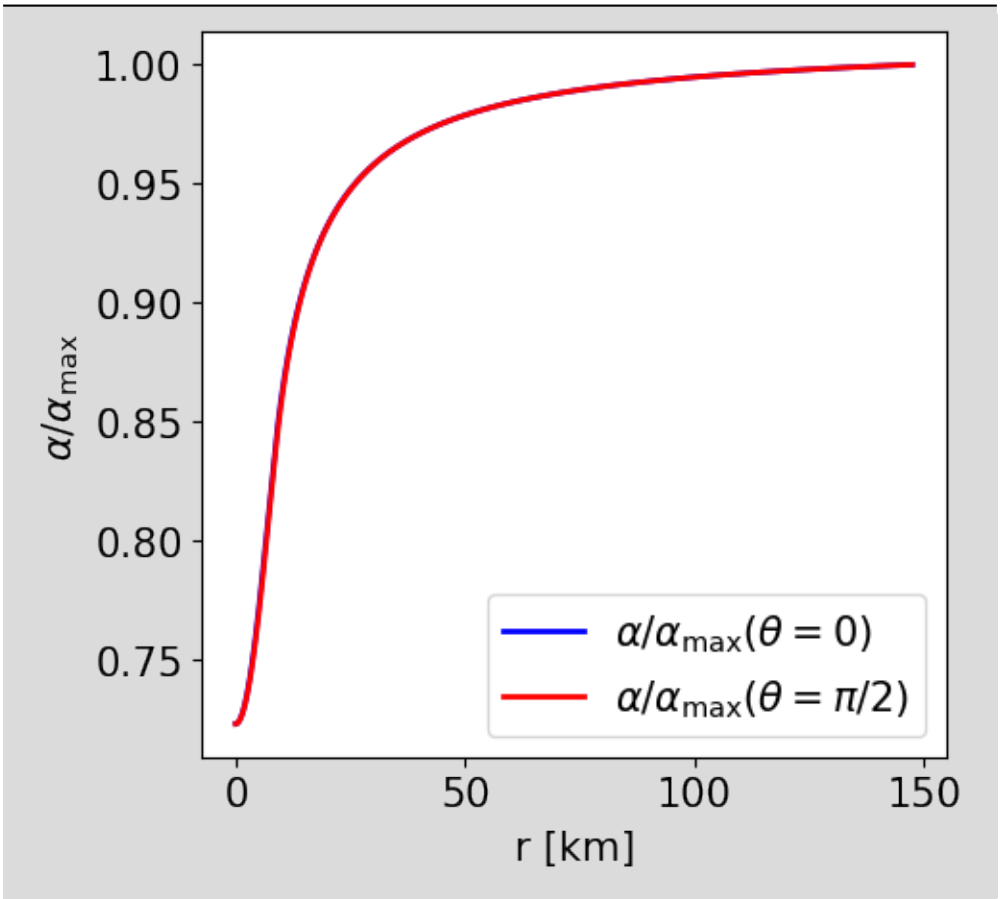
Radial profile of the NS density at the pole and at the equator.



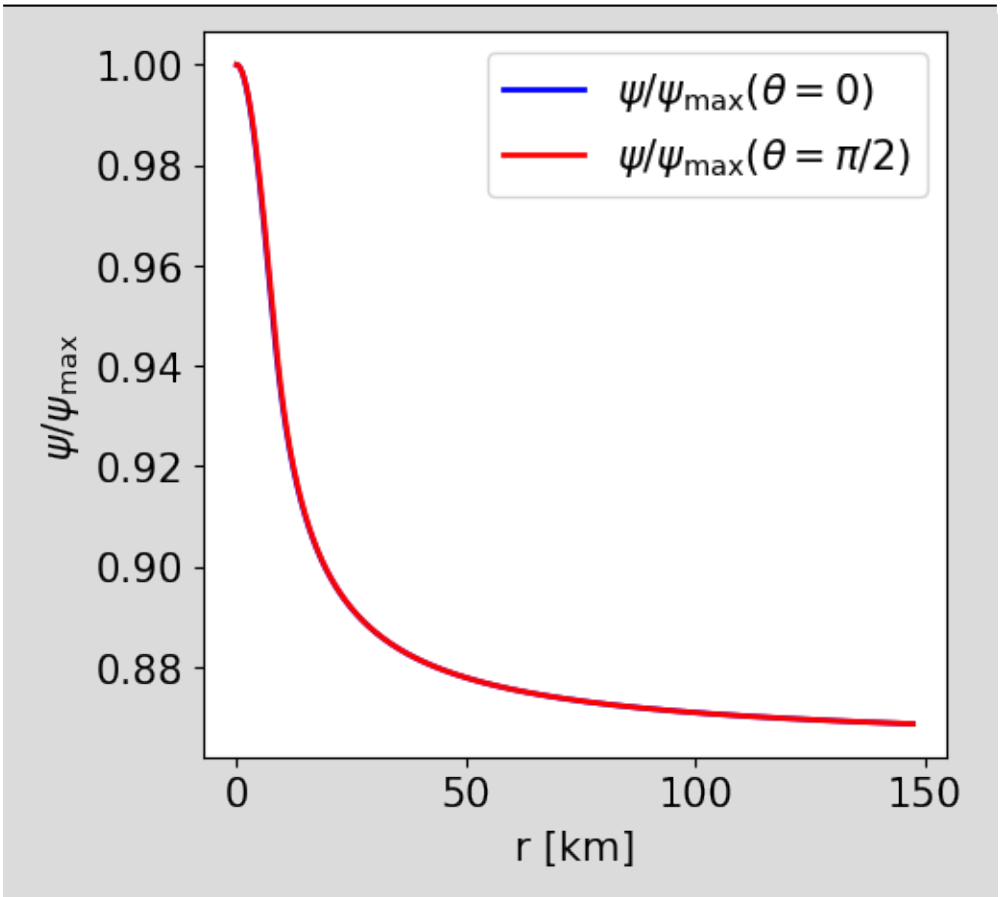
Radial profile of the NS pressure at the pole and at the equator.



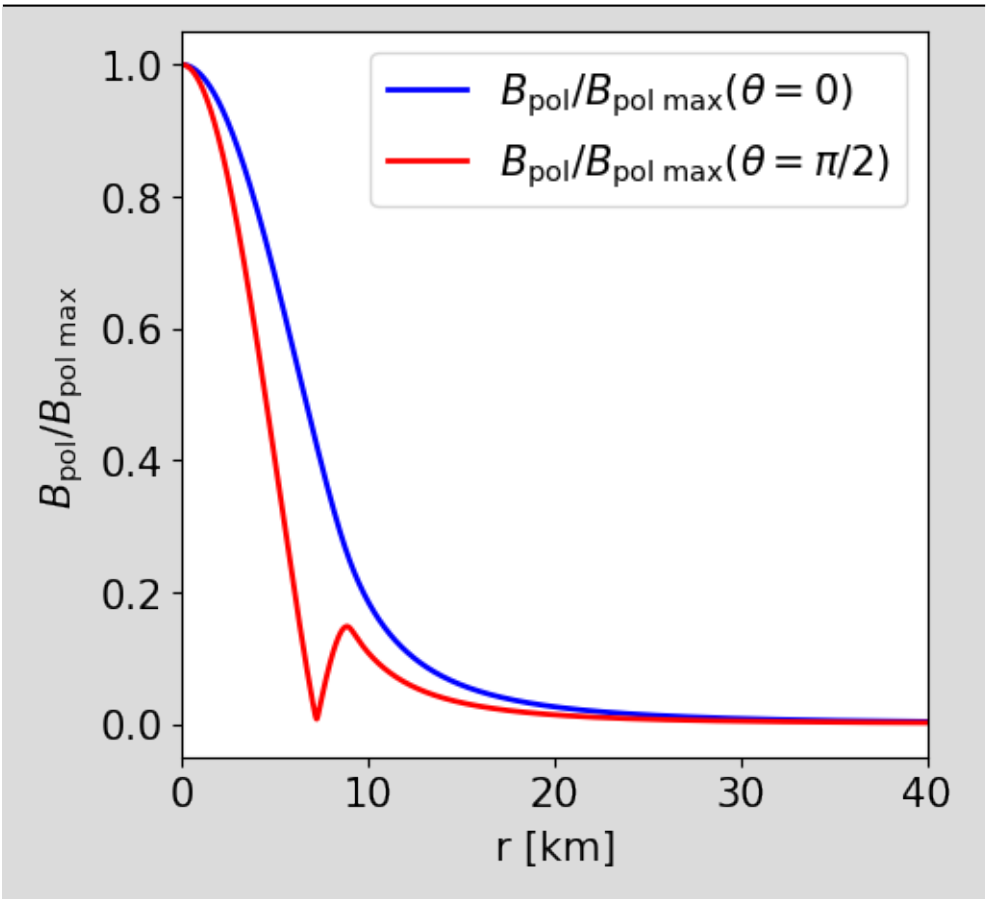
Radial profile of the scalar field at the pole and at the equator.



Radial profile of the lapse function at the pole and at the equator.



Radial profile of the conformal factor at the pole and at the equator.



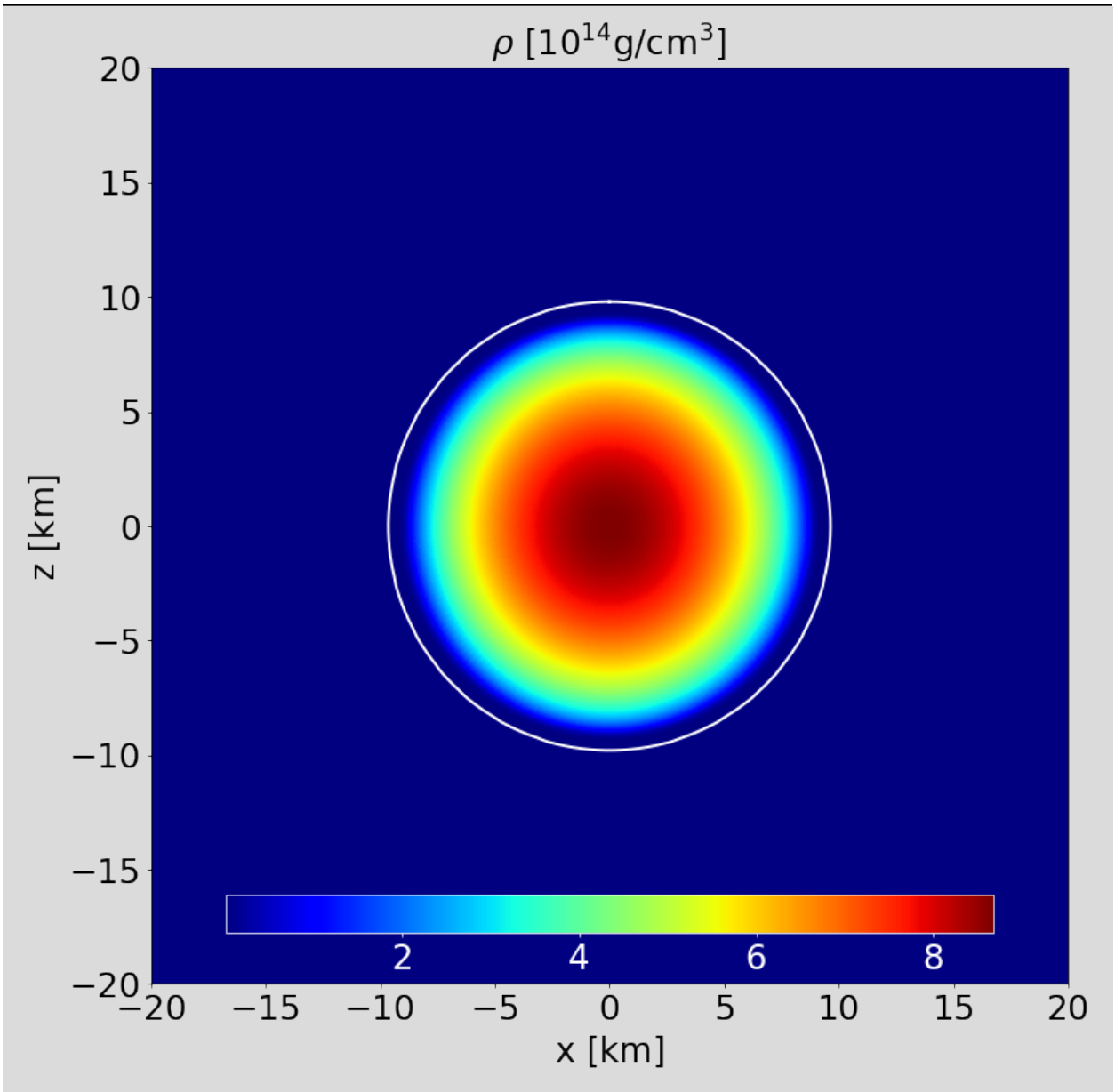
Radial profile of the poloidal magnetic field strength at the pole and at the equator.

7.3 Non-rotating NS in STT with the APR EoS and a purely toroidal magnetic field

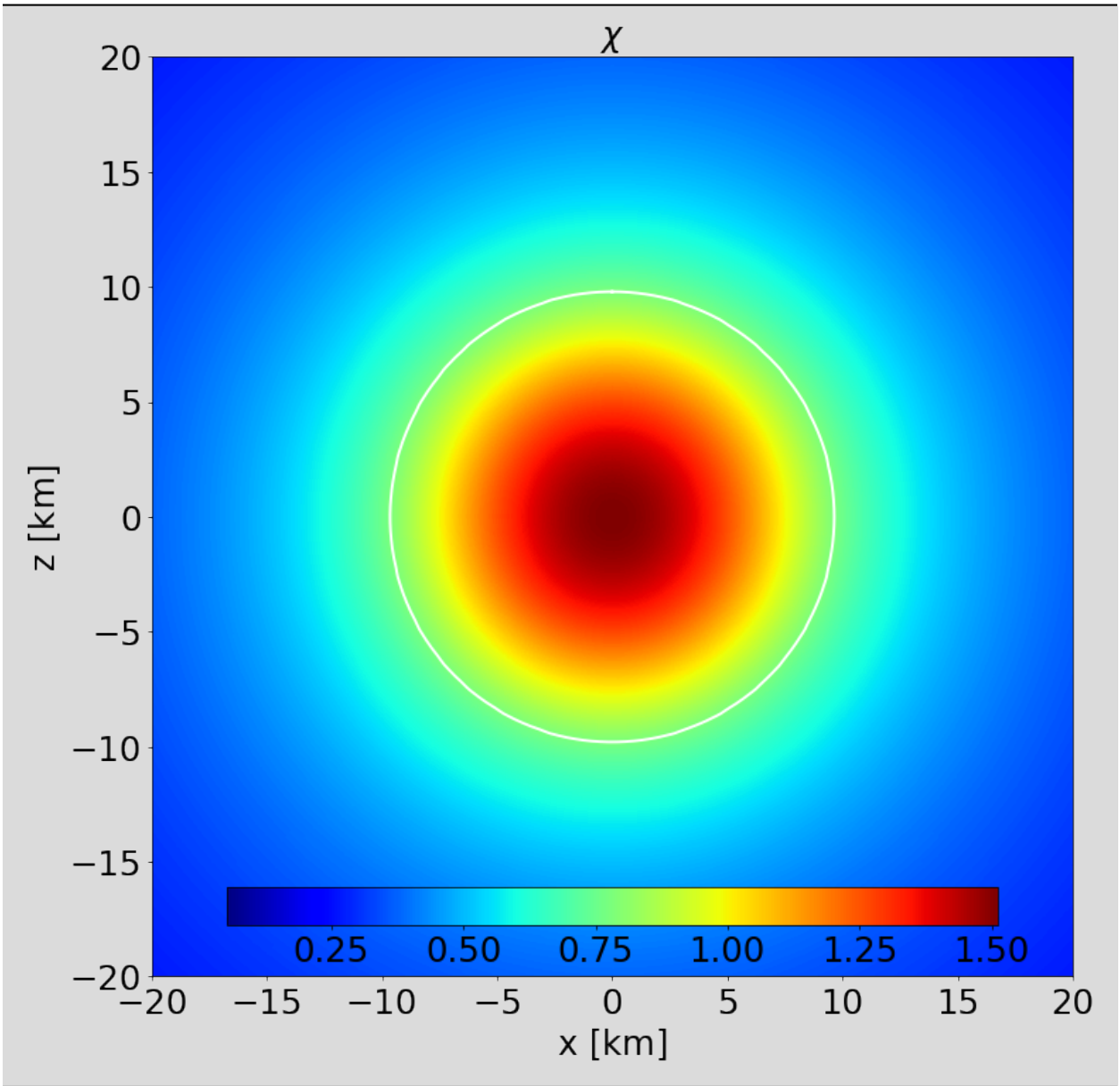
This is a model of an NS in STTs with $\alpha_0 = -2.0 \times 10^{-4}$ and $\beta_0 = -6$ described by the APR EoS, endowed with a purely toroidal field. It has a J-frame central density $\rho_c = 1.4 \times 10^{-3}$ in code units (corresponding to 8.684×10^{14} gcm^{-3}) and a Komar mass in the E-frame of 1.078 M$_\odot$. The circumferential radius in the J-frame is 11.060 km, and the E-frame scalar charge is 0.515 M$_\odot$.

Here are the main parameters used to compute this model. The detailed log with all the parameters can be found in the corresponding LogFile.dat.

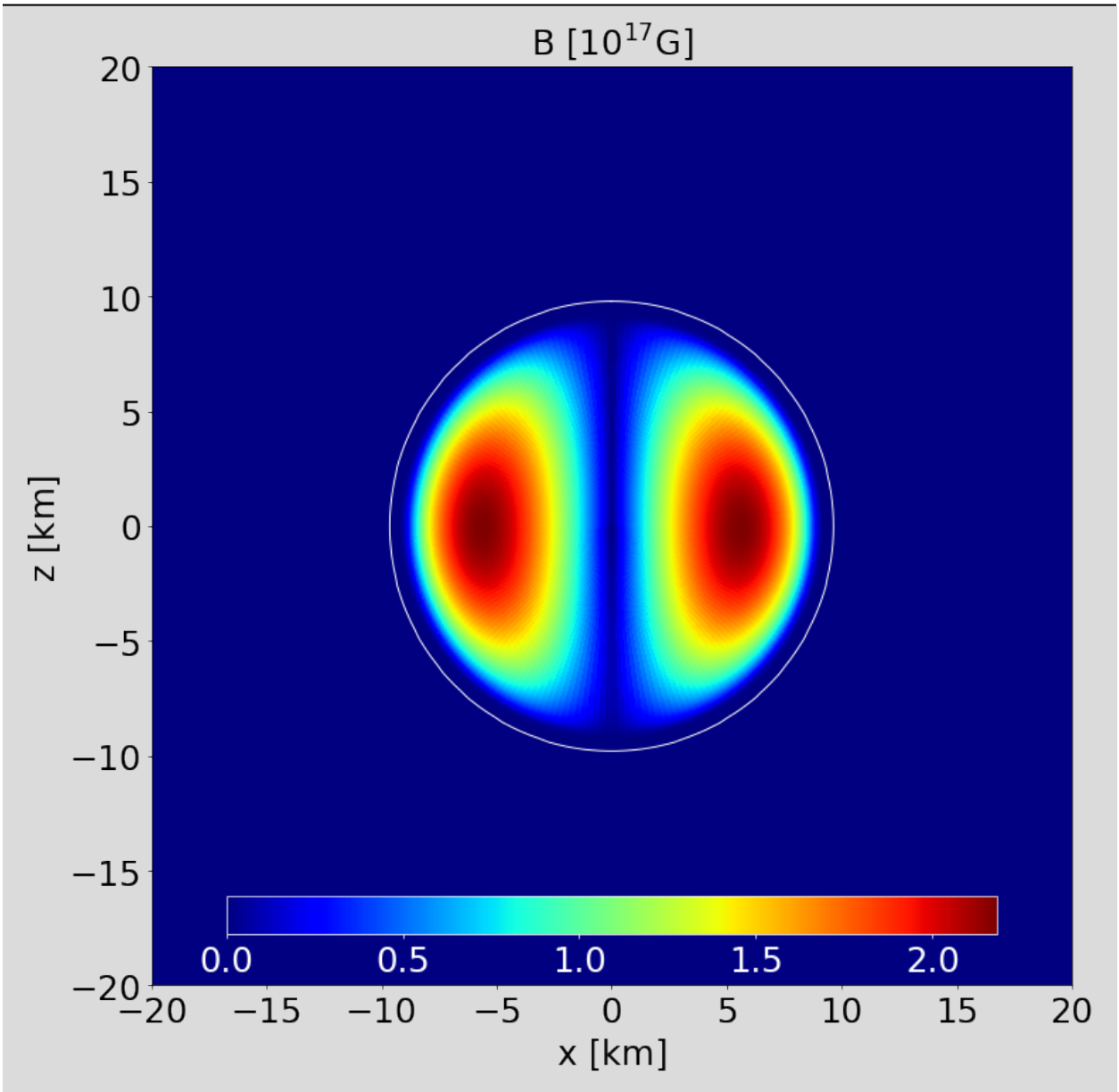
```
NR = 900, NTH = 100, NRREG = 600, MLS = 20, NGQ = 50, RREG = 10, RMAXSTR = 100,
RMAX = 100, REQMAX = 11.50, RHOINI = 1.40E-3, ALPHA0 = -2.0E-4, BETA0 = -6, GR = .FALSE.,
STRETCH = .TRUE., ANALYTIC = .TRUE., CONVHELP = .FALSE., QFACTOR = 0.85, QFACTORCHI = 0.
↪45,
QFACTORMETRIC = 0.35, QRELAX = 0.30, QAPHI = 0.50, EOSINT = .TRUE., FILEEOS = 'APR_
↪resampled.dat',
IMAG = .TRUE., ITOR = .TRUE., BCOEF = 0.7, NPOL = 0.0, CSI = 0.0
```



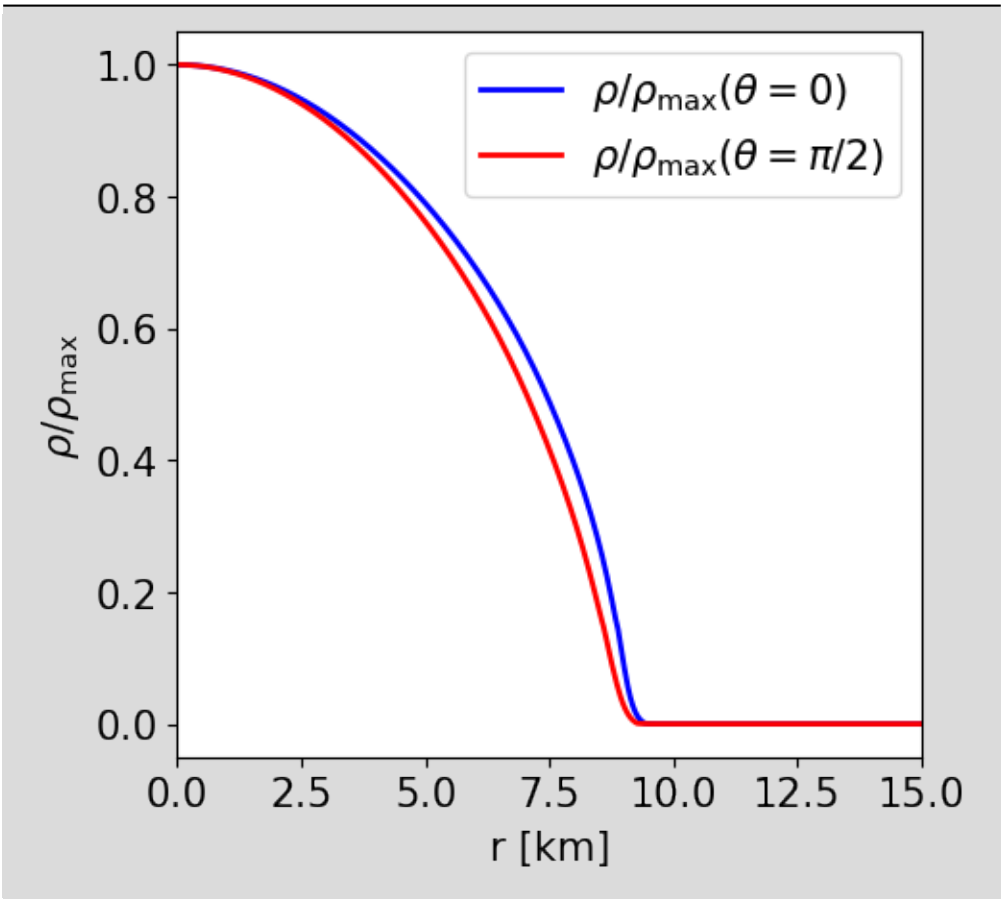
Section of the NS in the $x-z$ plane along with the contours the density.



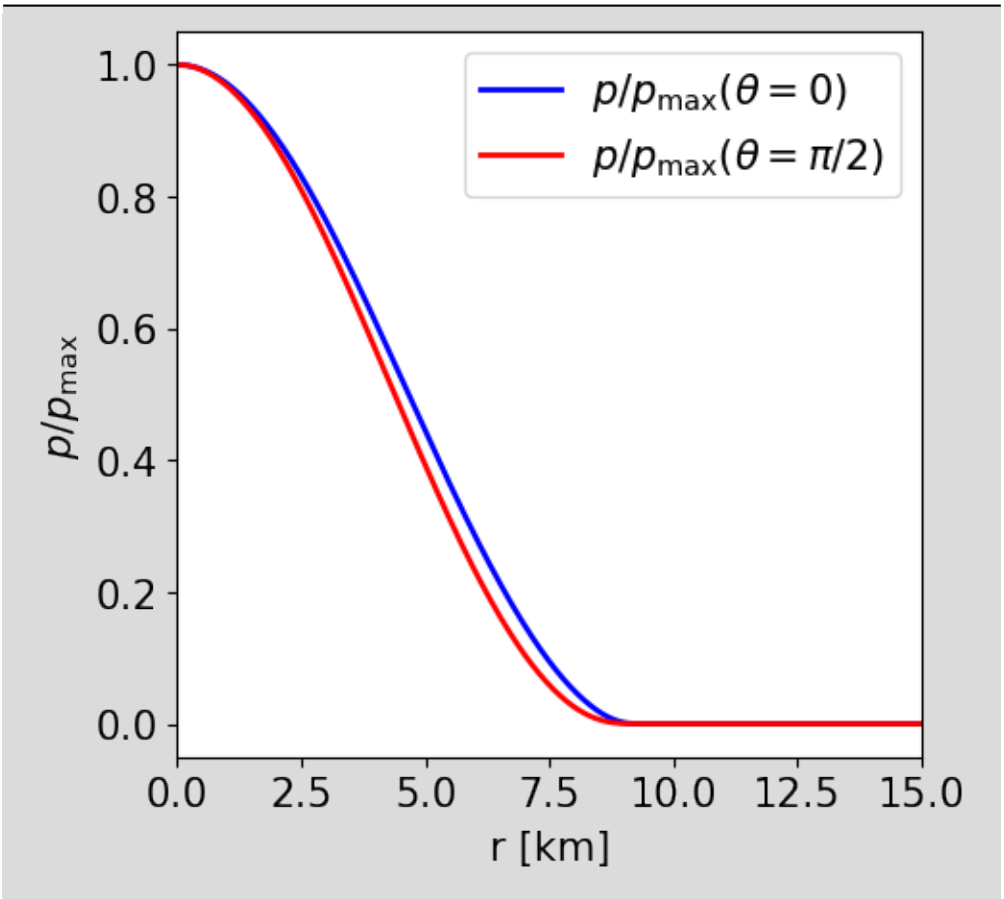
Section of the NS in the $x - z$ plane along with the contours the scalar field.



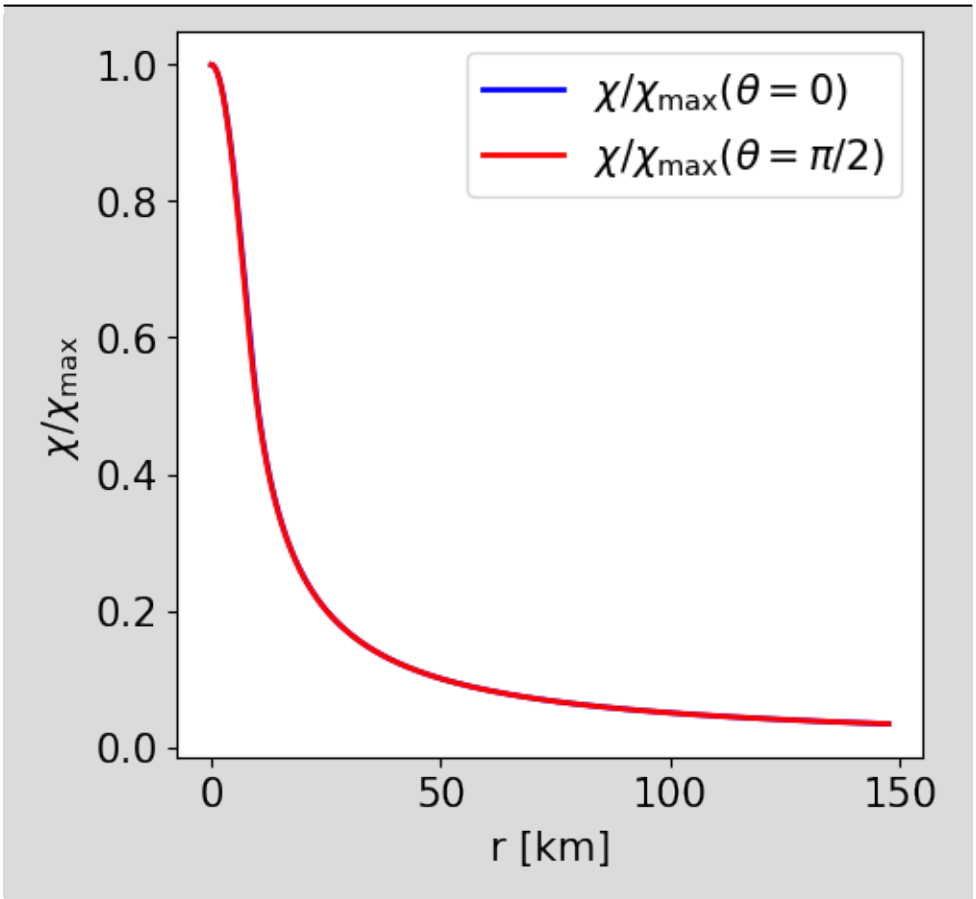
Section of the NS in the $x - z$ plane along with the contours the toroidal magnetic field strength.



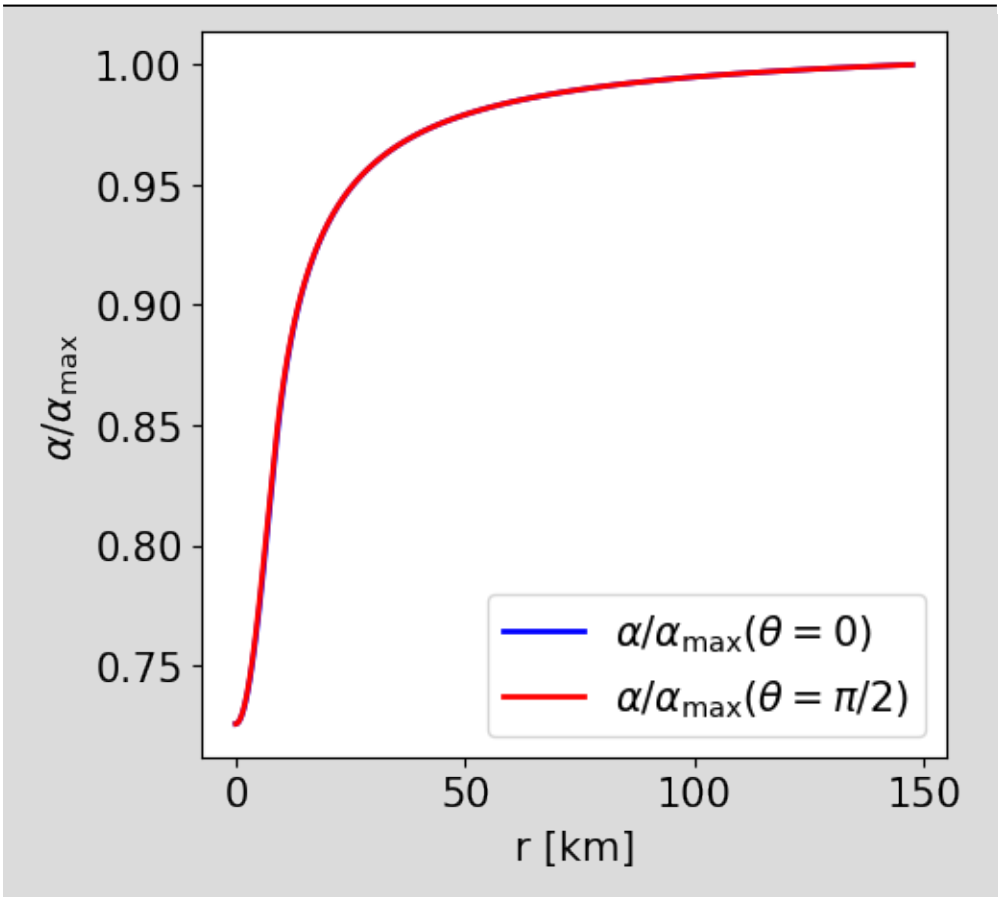
Radial profile of the NS density at the pole and at the equator.



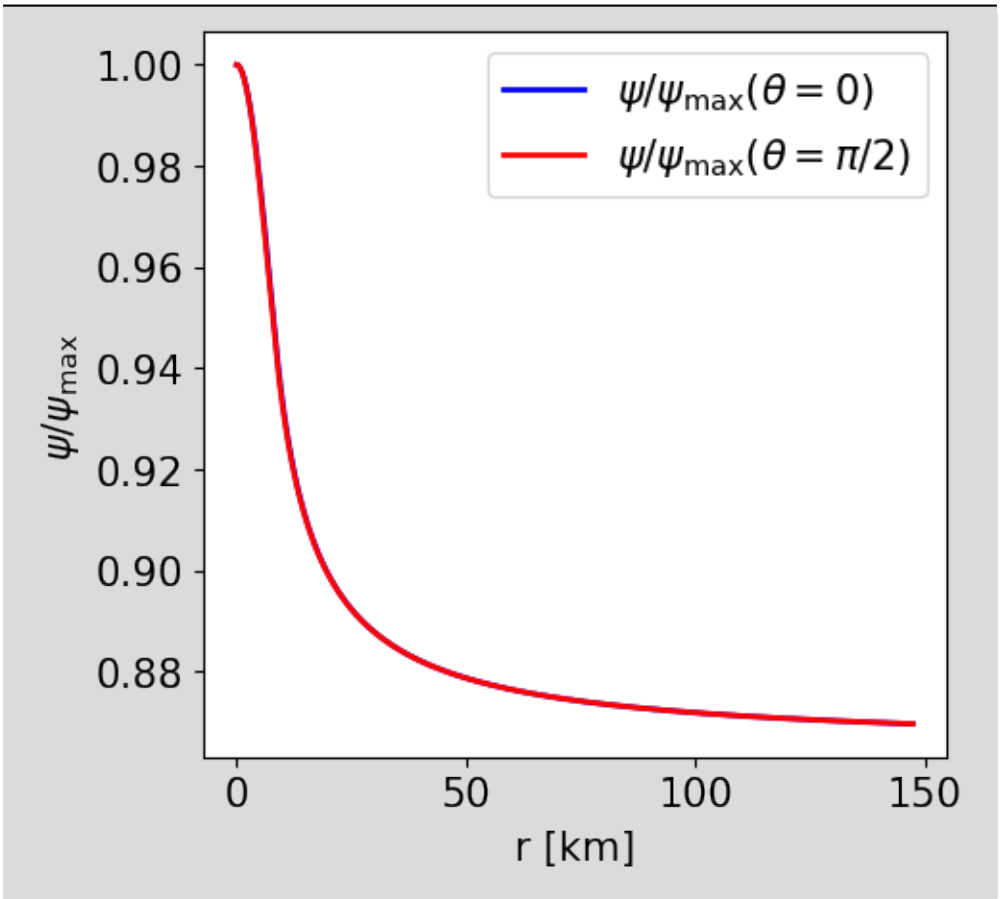
Radial profile of the NS pressure at the pole and at the equator.



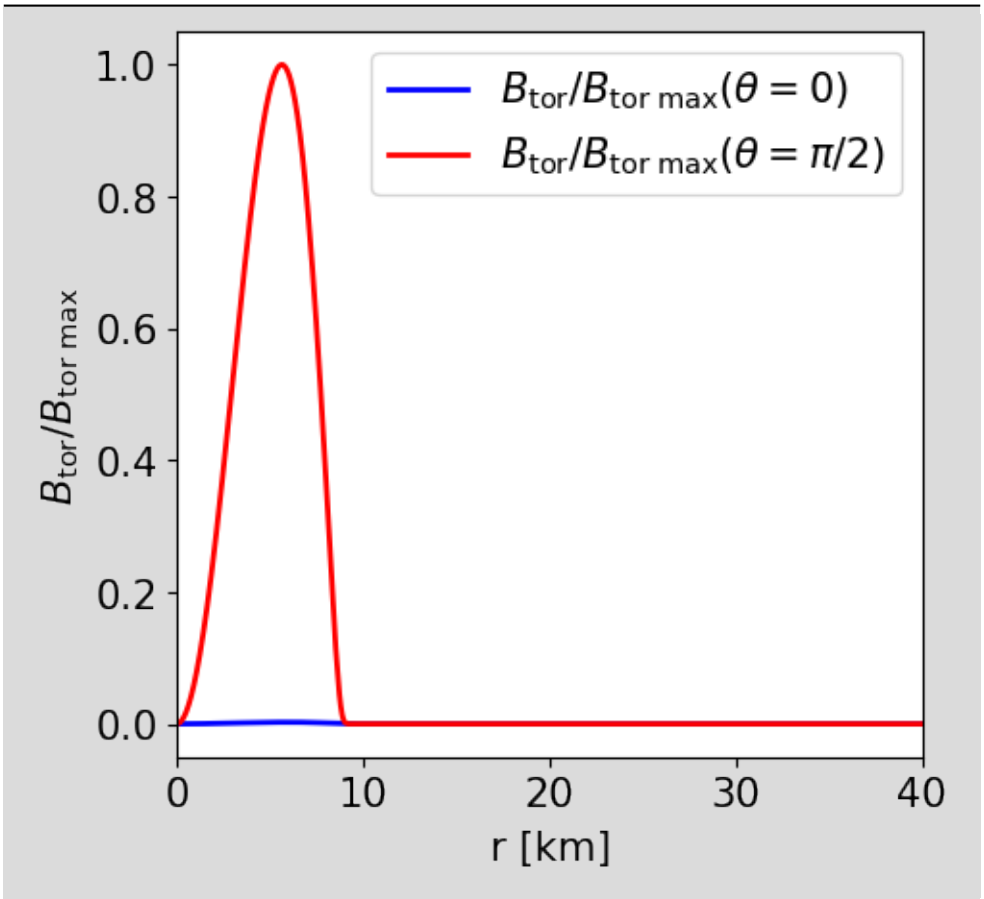
Radial profile of the scalar field at the pole and at the equator.



Radial profile of the lapse function at the pole and at the equator.



Radial profile of the conformal factor at the pole and at the equator.



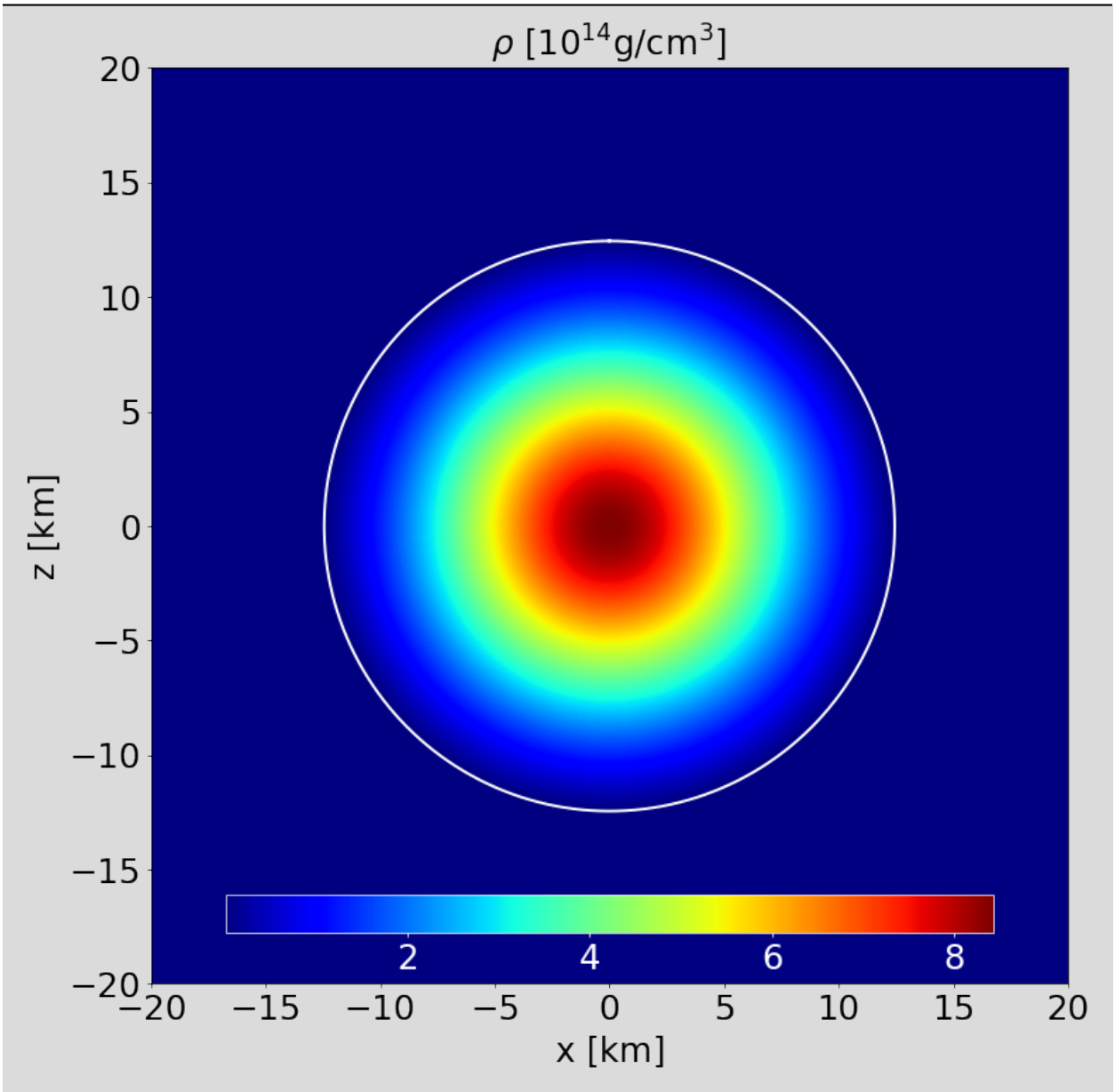
Radial profile of the toroidal magnetic field strength at the pole and at the equator.

7.4 Non-rotating, unmagnetised NS in STT with the POL2 EoS

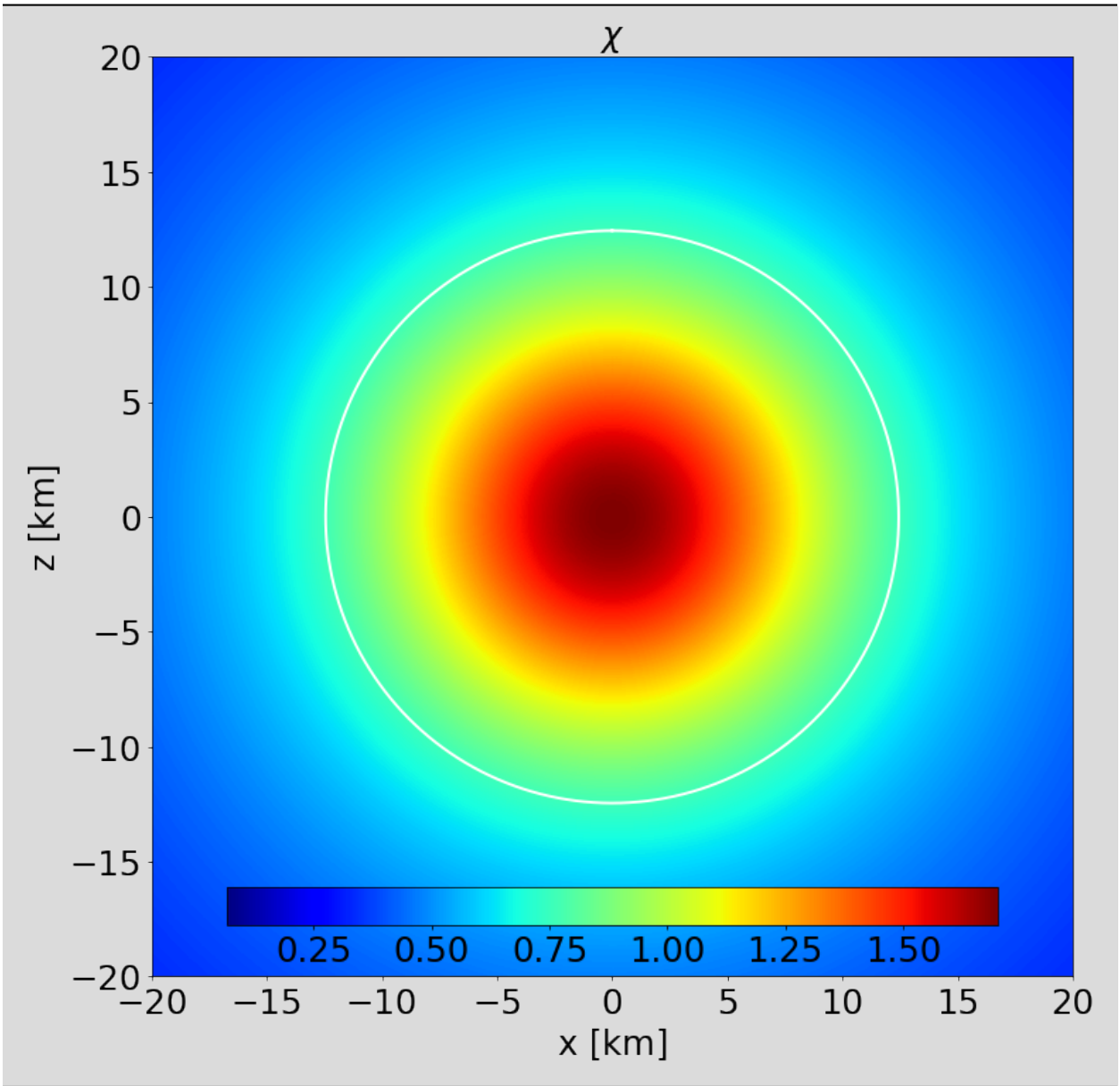
This is a model of an unmagnetised NS in STTs with $\alpha_0 = -2.0 \times 10^{-4}$ and $\beta_0 = -6$ described by the analytic POL2 EoS. It has a J-frame central density $\rho_c = 1.36 \times 10^{-3}$ in code units (corresponding to 8.44×10^{14} gcm^{-3}) and a Komar mass in the E-frame of 1.296 M$_\odot$. The circumferential radius in the J-frame is 14.156 km, and the E-frame scalar charge is 0.645 M$_\odot$.

Here are the main parameters used to compute this model. The detailed log with all the parameters can be found in the corresponding LogFile.dat.

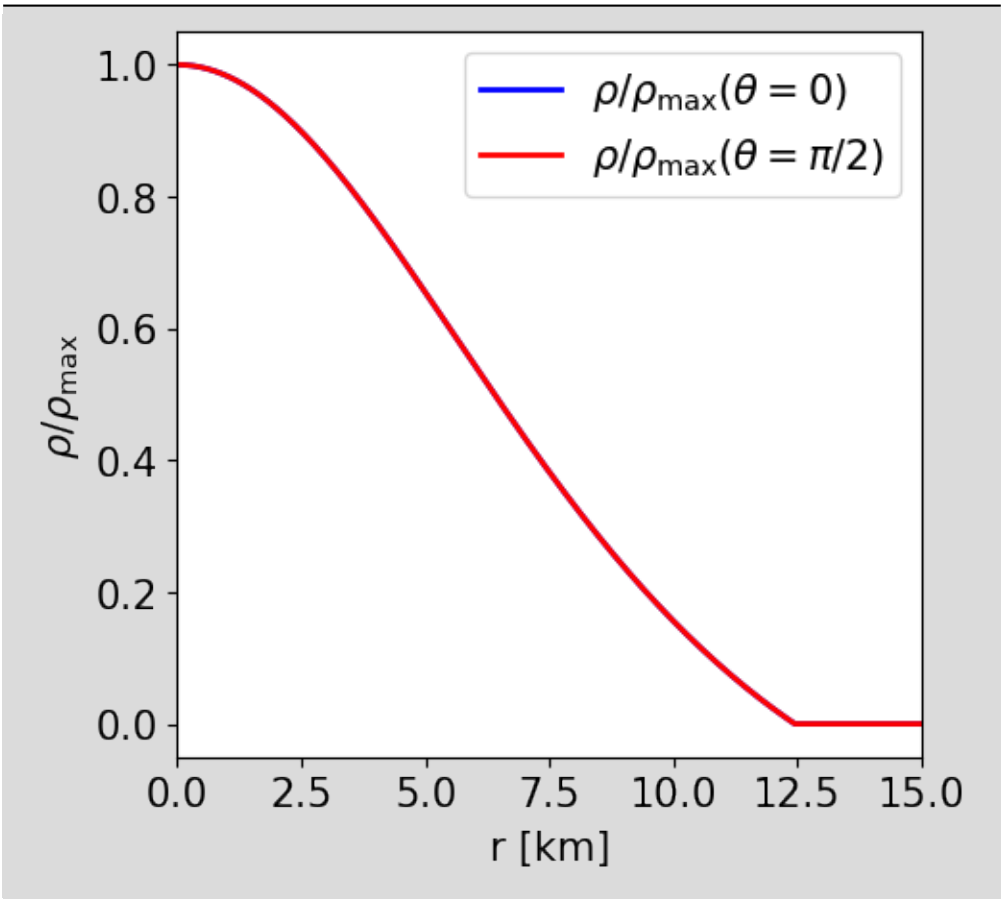
```
NR = 900, NTH = 100, NRREG = 600, MLS = 20, NGQ = 50, RREG = 10, RMAXSTR = 100,
RMAX = 100, REQMAX = 15.0, RHOINI = 1.36E-3, ALPHA0 = -2.0E-4, BETA0 = -6, GR = .FALSE.,
STRETCH = .TRUE., ANALYTIC = .TRUE., CONVHELP = .FALSE., QFACTOR = 0.85, QFACTORCHI = 0.
↪45,
QFACTORMETRIC = 0.35, QRELAX = 0.30, QAPHI = 0.50, EOSINT = .FALSE., K1 = 110.0, GAMMA =␣
↪2.0
```



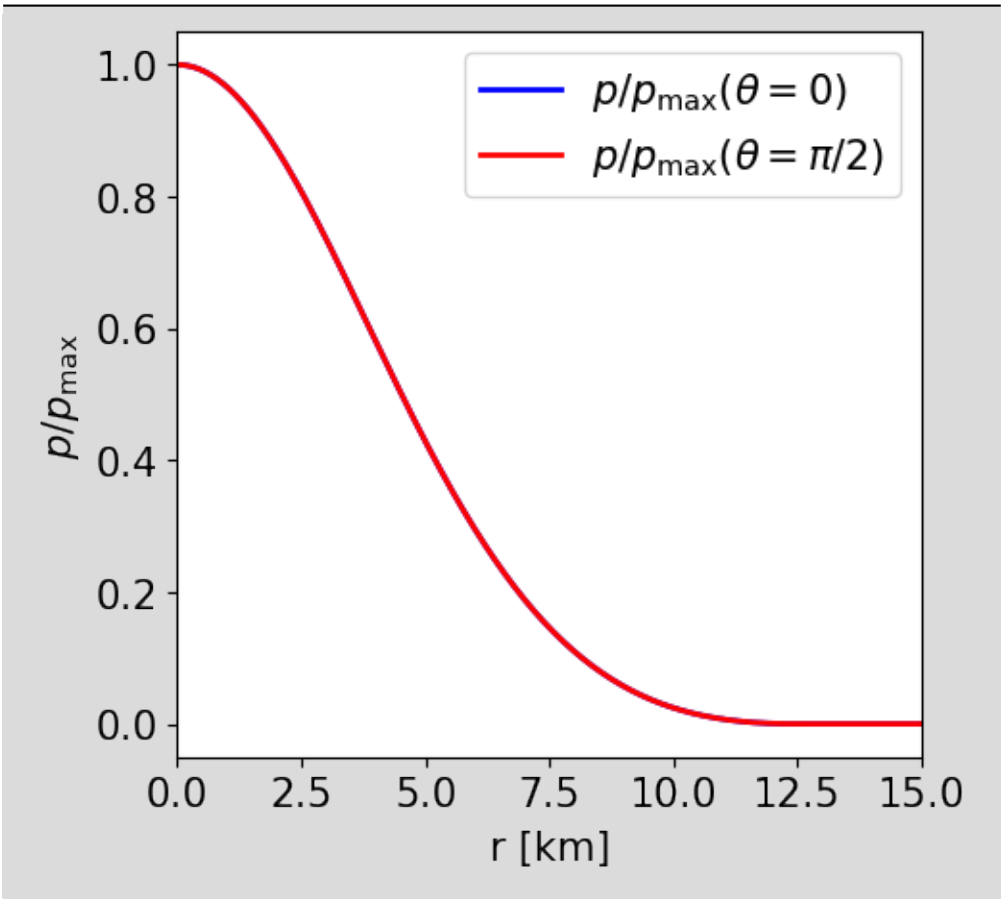
Section of the NS in the $x - z$ plane along with the contours the density.



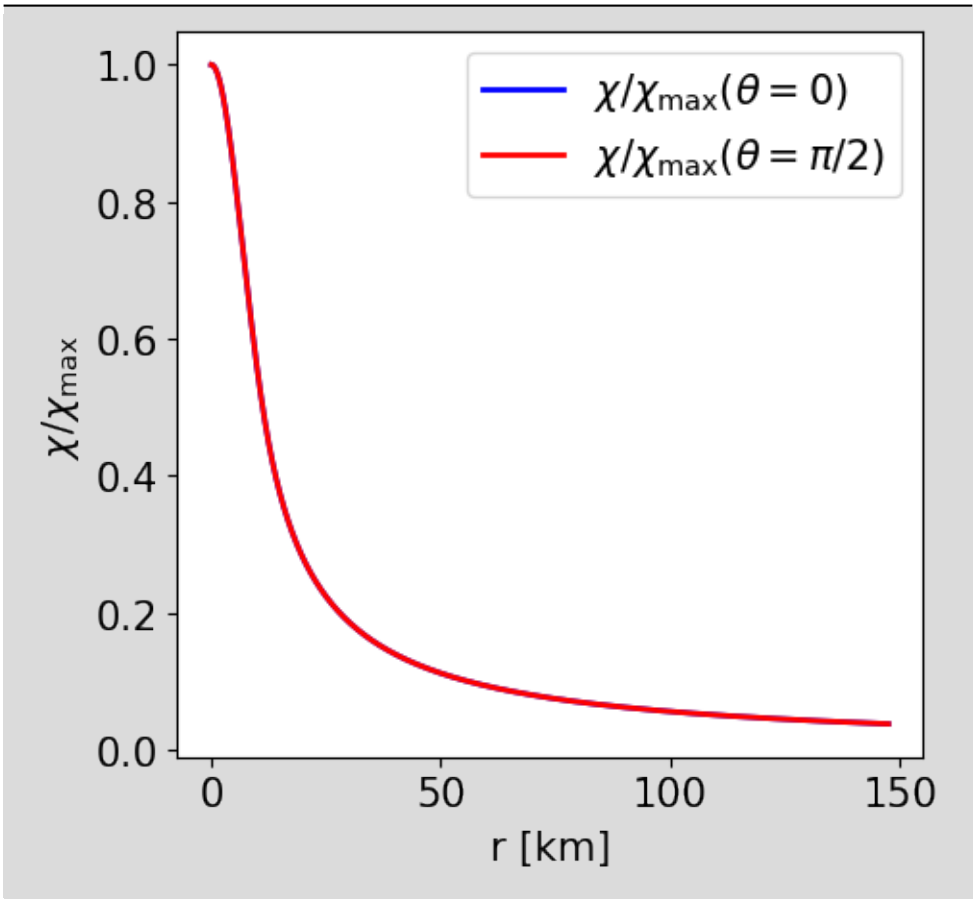
Section of the NS in the $x - z$ plane along with the contours the scalar field.



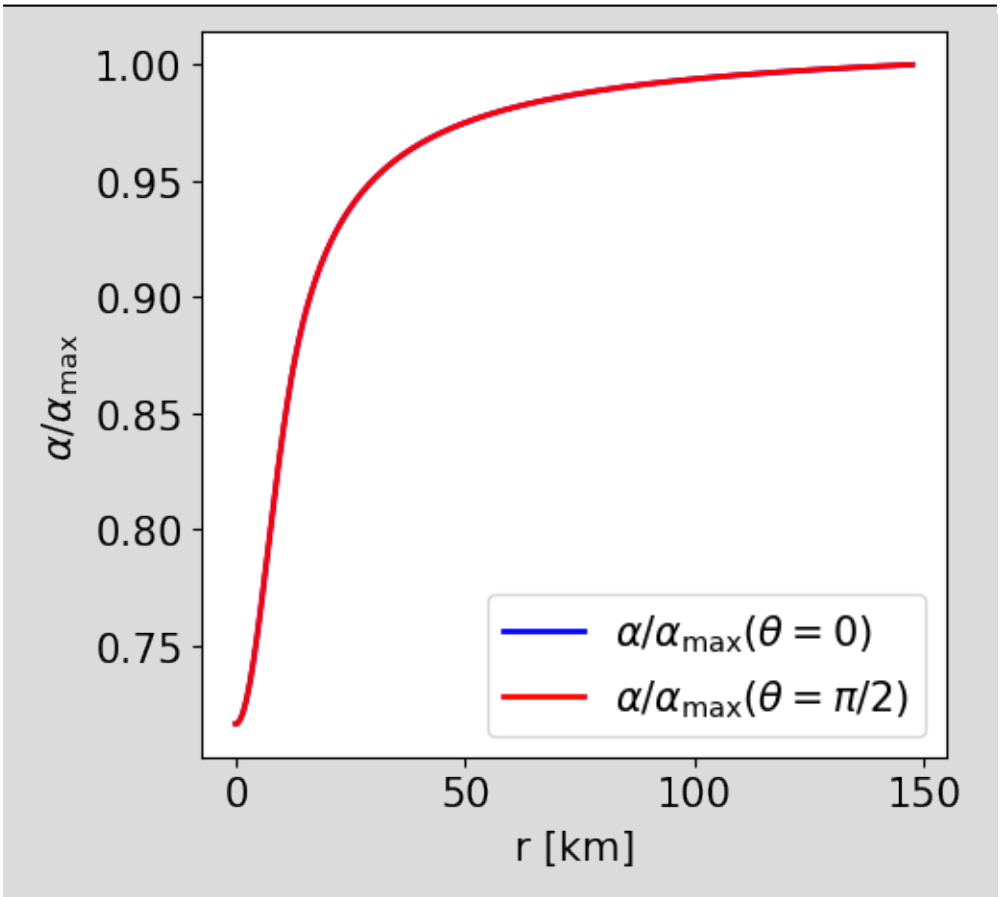
Radial profile of the NS density at the pole and at the equator.



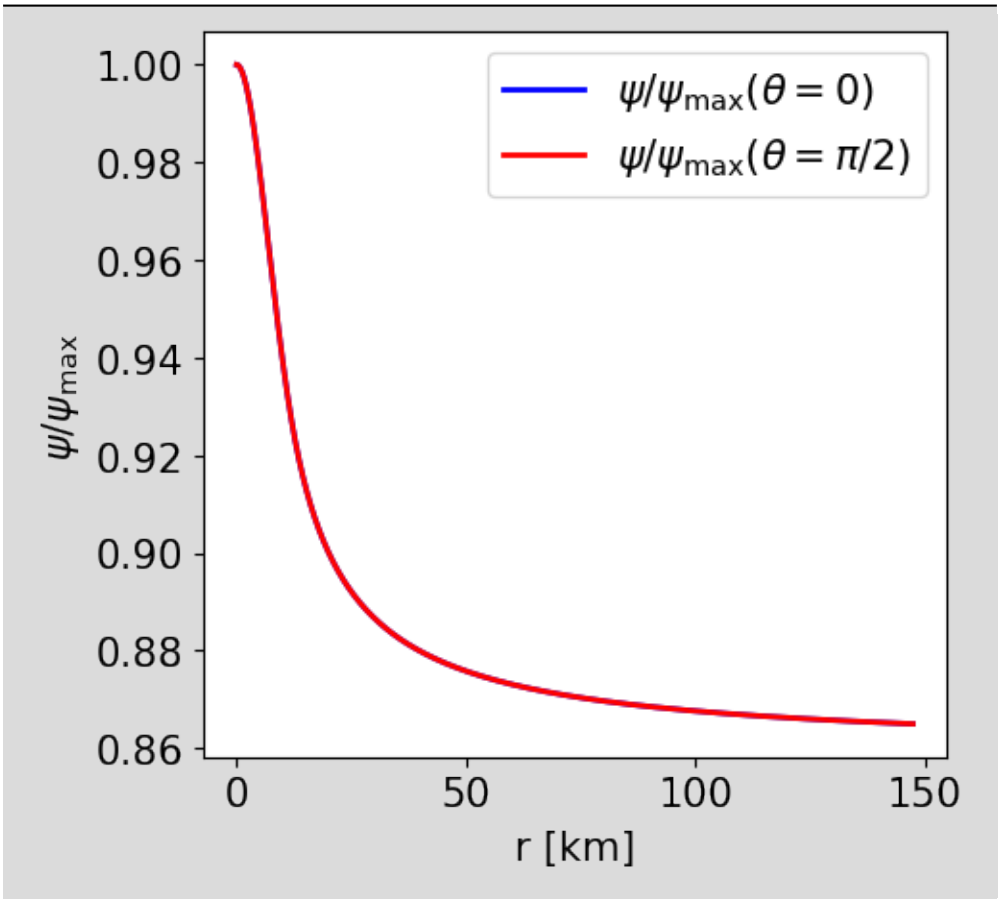
Radial profile of the NS pressure at the pole and at the equator.



Radial profile of the scalar field at the pole and at the equator.



Radial profile of the lapse function at the pole and at the equator.



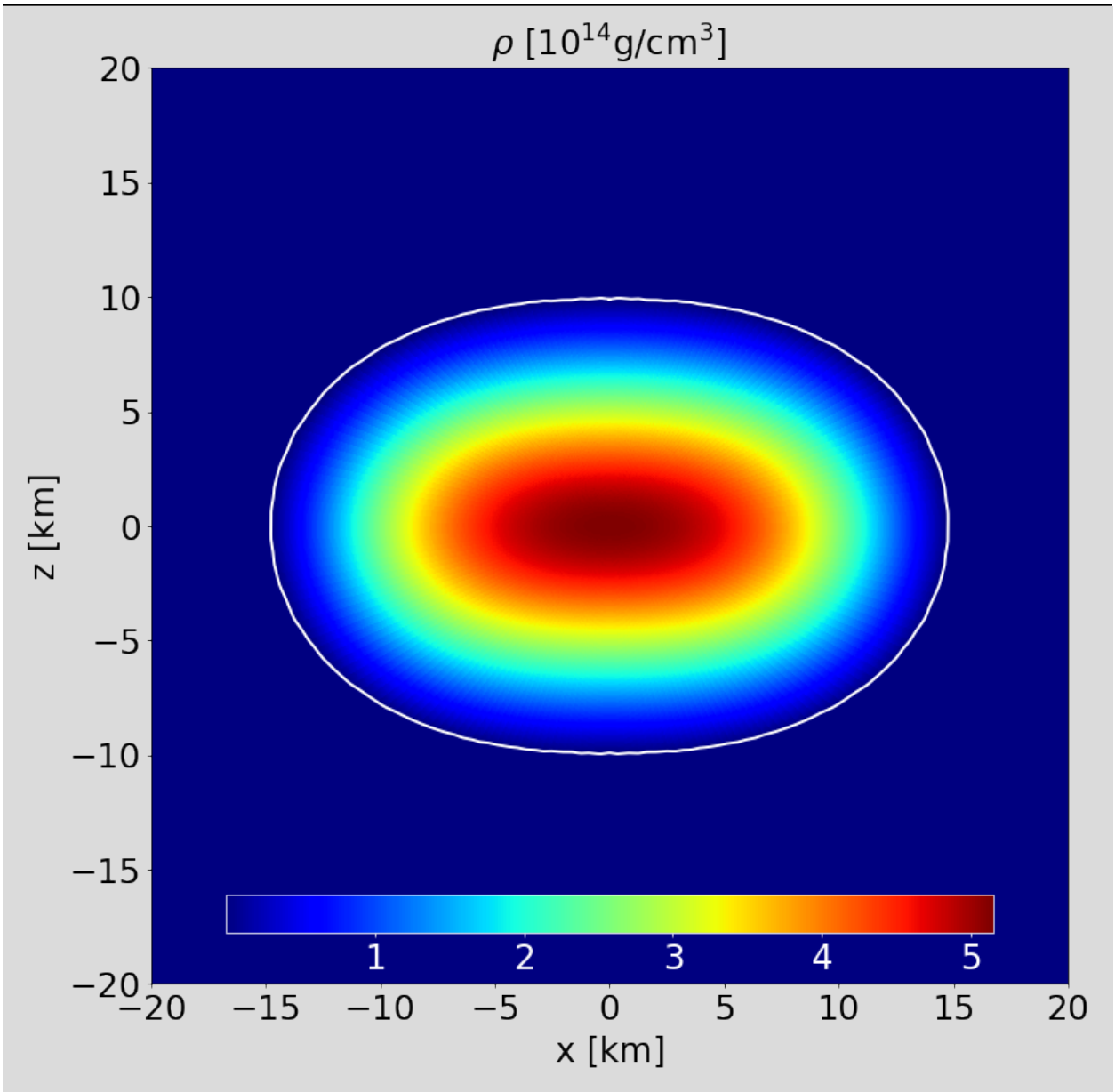
Radial profile of the conformal factor at the pole and at the equator.

7.5 Non-rotating NS in STT with the POL2 EoS and a purely poloidal magnetic field

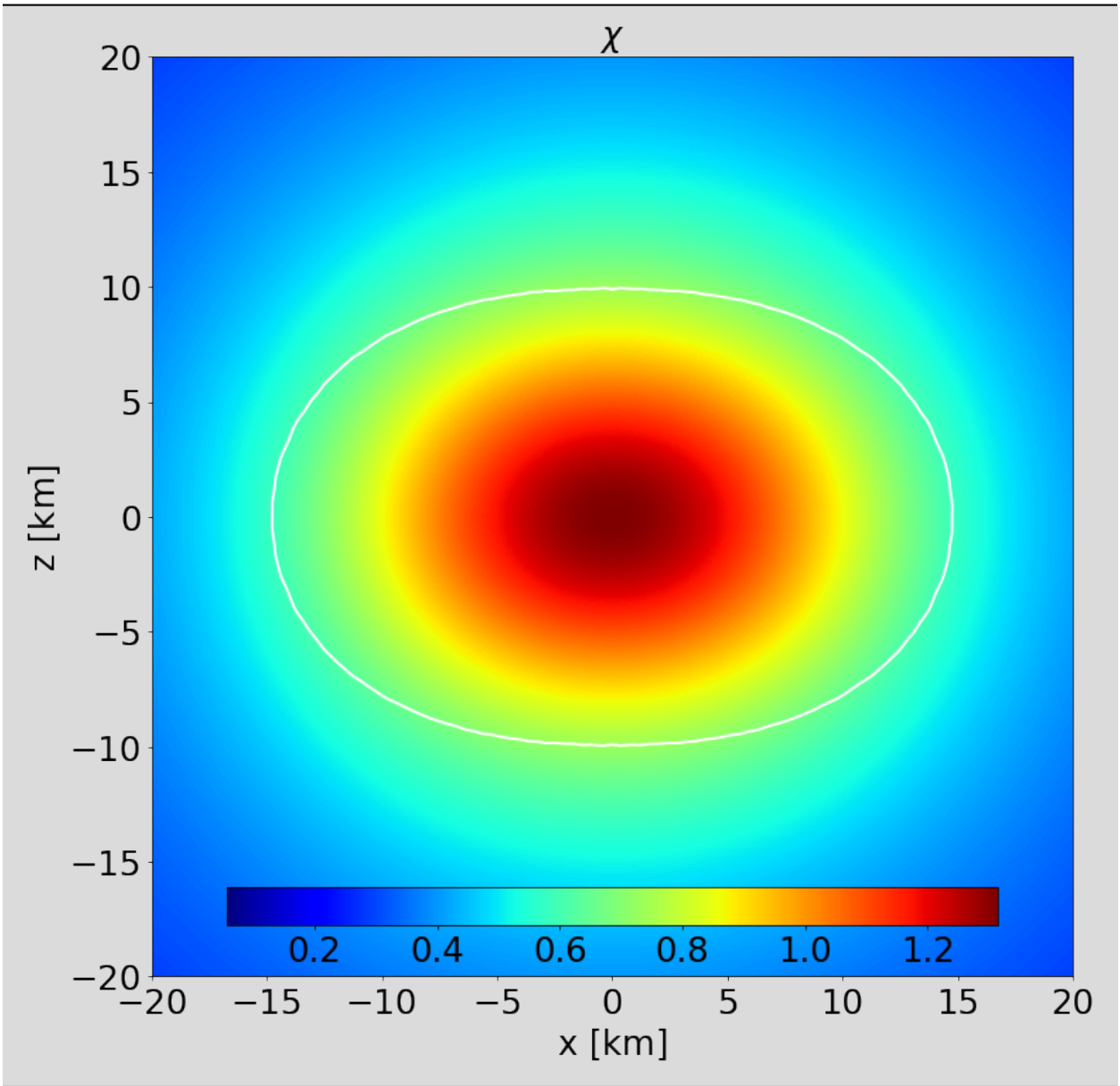
This is a model of an NS in STTs with $\alpha_0 = -2.0 \times 10^{-4}$ and $\beta_0 = -6$ described by the analytic POL2 EoS, endowed with a purely poloidal field. It has a J-frame central density $\rho_c = 8.30 \times 10^{-4}$ in code units (corresponding to 5.15×10^{14} gcm^{-3}) and a Komar mass in the E-frame of 1.356 M$_\odot$. The circumferential radius in the J-frame is 16.718 km, and the E-frame scalar charge is 0.564 M$_\odot$.

Here are the main parameters used to compute this model. The detailed log with all the parameters can be found in the corresponding LogFile.dat.

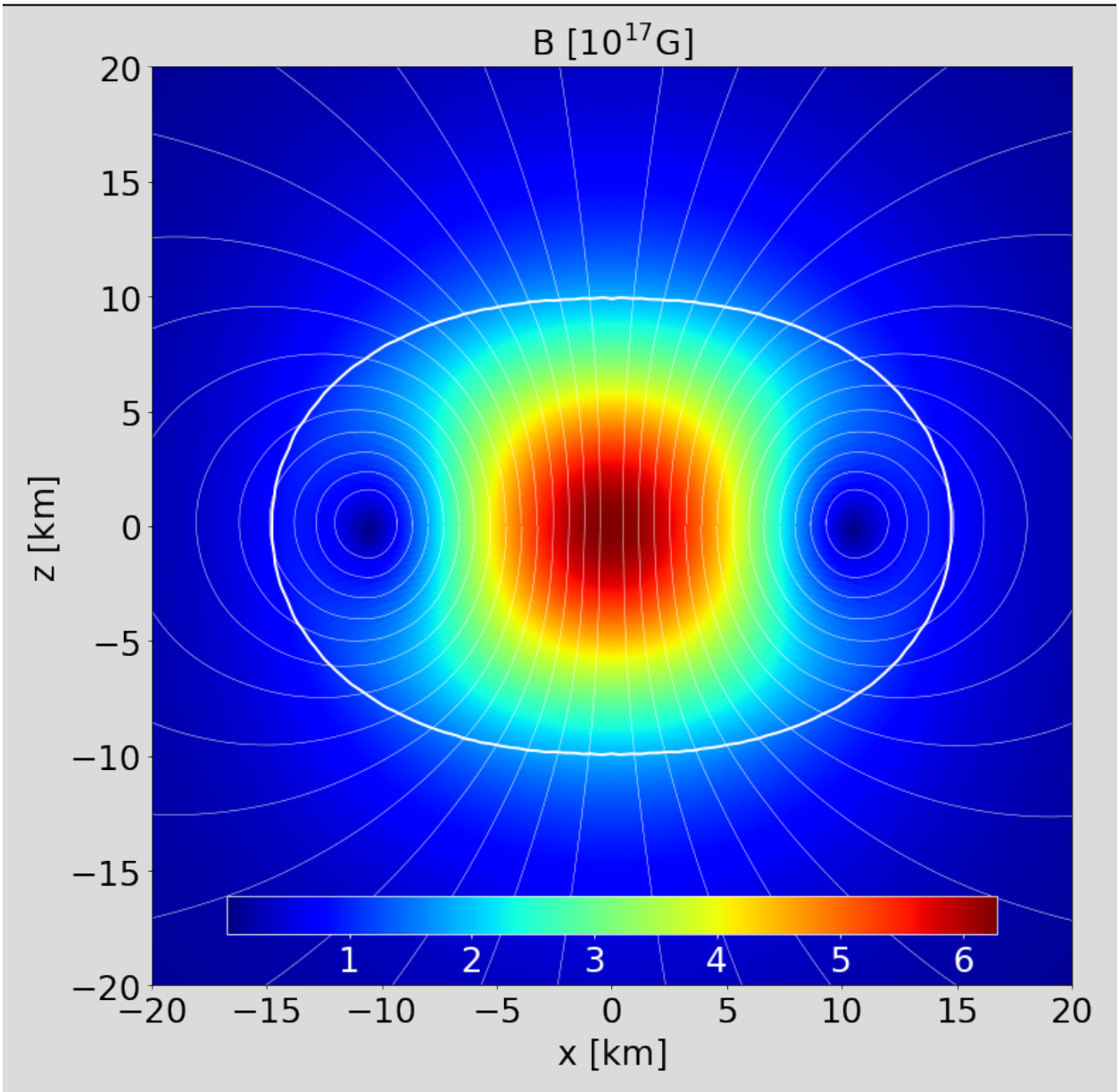
```
NR = 900, NTH = 100, NRREG = 600, MLS = 20, NGQ = 50, RREG = 20, RMAXSTR = 100,
RMAX = 100, REQMAX = 25.0, RHOINI = 8.30E-4, ALPHA0 = -2.0E-4, BETA0 = -6, GR = .FALSE.,
STRETCH = .TRUE., ANALYTIC = .TRUE., CONVHELP = .FALSE., QFACTOR = 0.85, QFACTORCHI = 0.
↪45,
QFACTORMETRIC = 0.35, QRELAX = 0.30, QAPHI = 0.50, EOSINT = .FALSE., K1 = 110.0, GAMMA =␣
↪2.0,
IMAG = .TRUE., IPOL = .TRUE., KBPOL = 0.44, NPOL = 0.0, CSI = 0.0
```



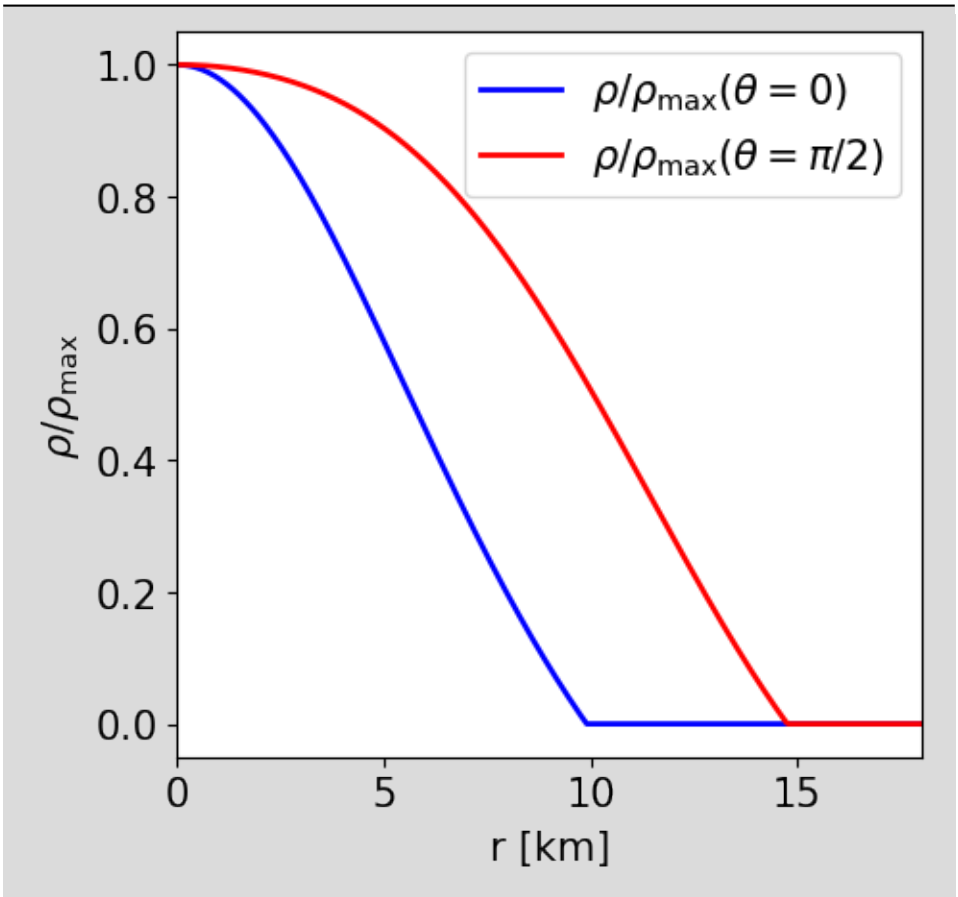
Section of the NS in the $x - z$ plane along with the contours the density.



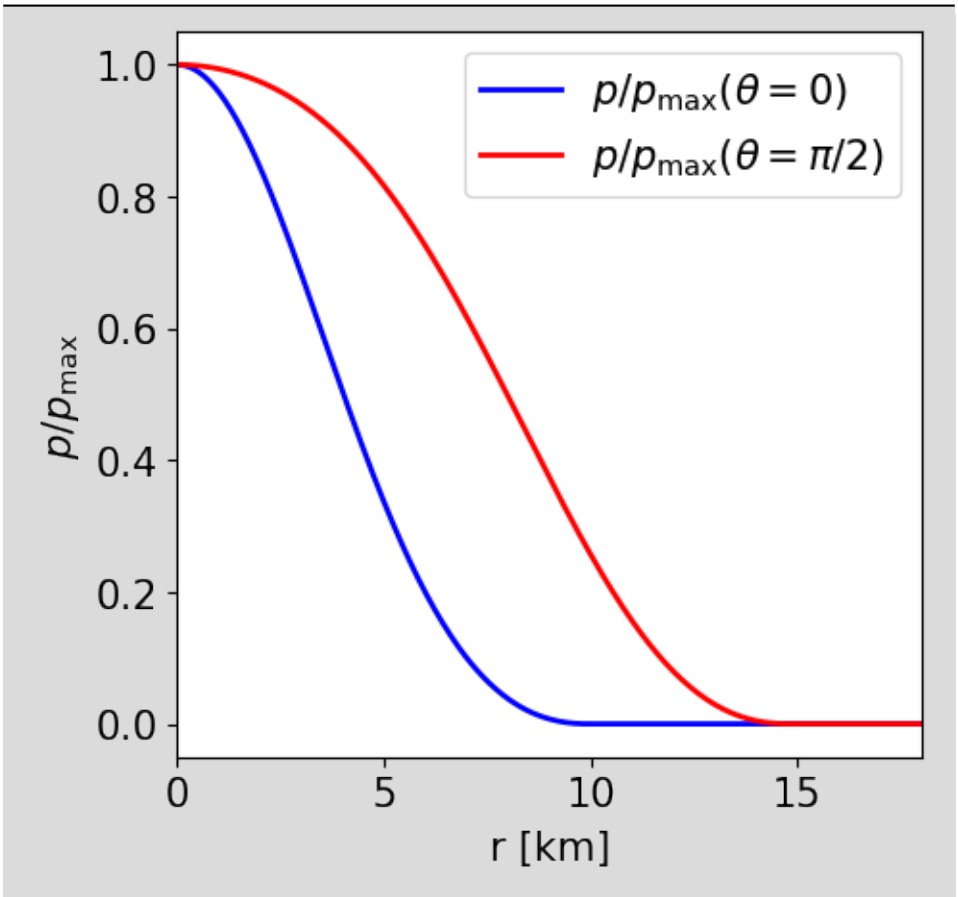
Section of the NS in the $x - z$ plane along with the contours the scalar field.



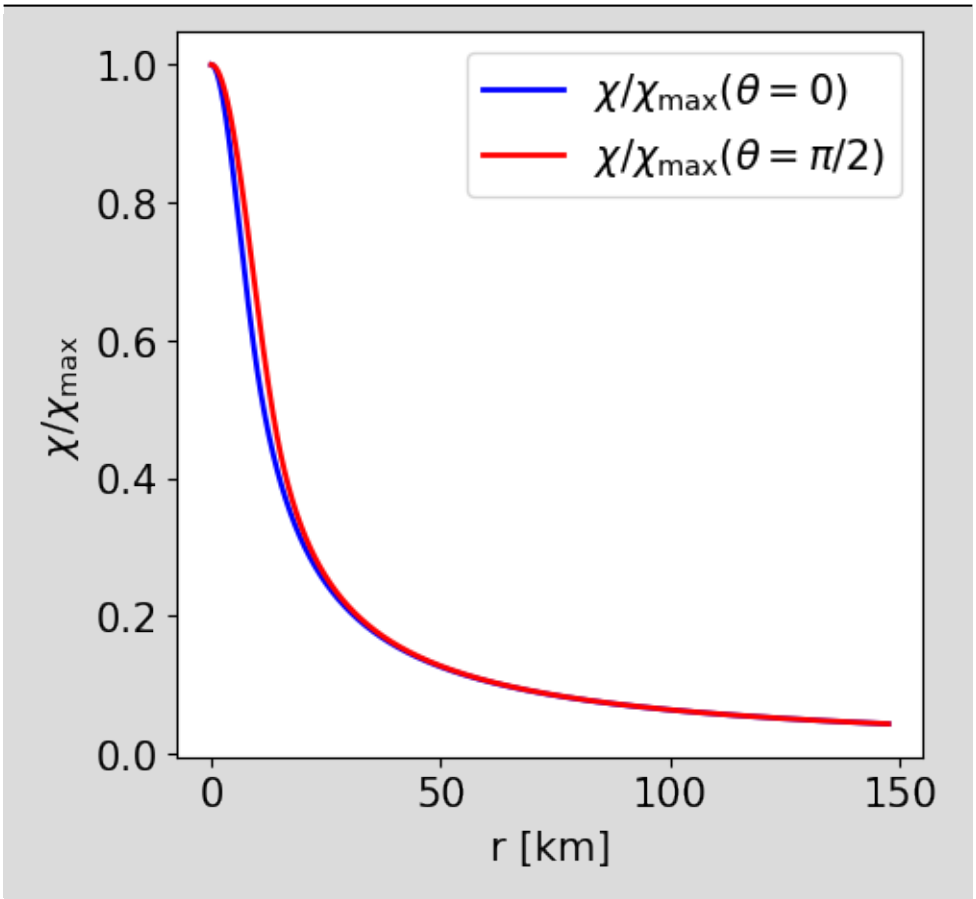
Section of the NS in the $x - z$ plane along with the contours the poloidal magnetic field strength and its field lines.



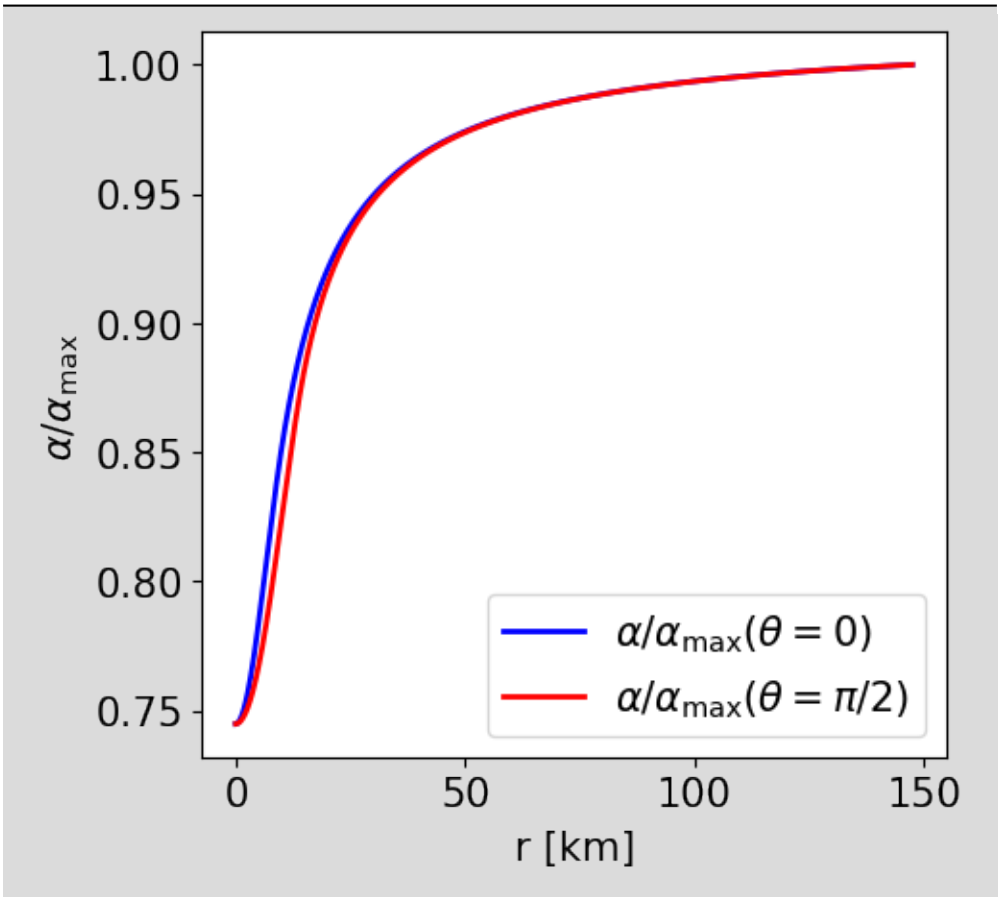
Radial profile of the NS density at the pole and at the equator.



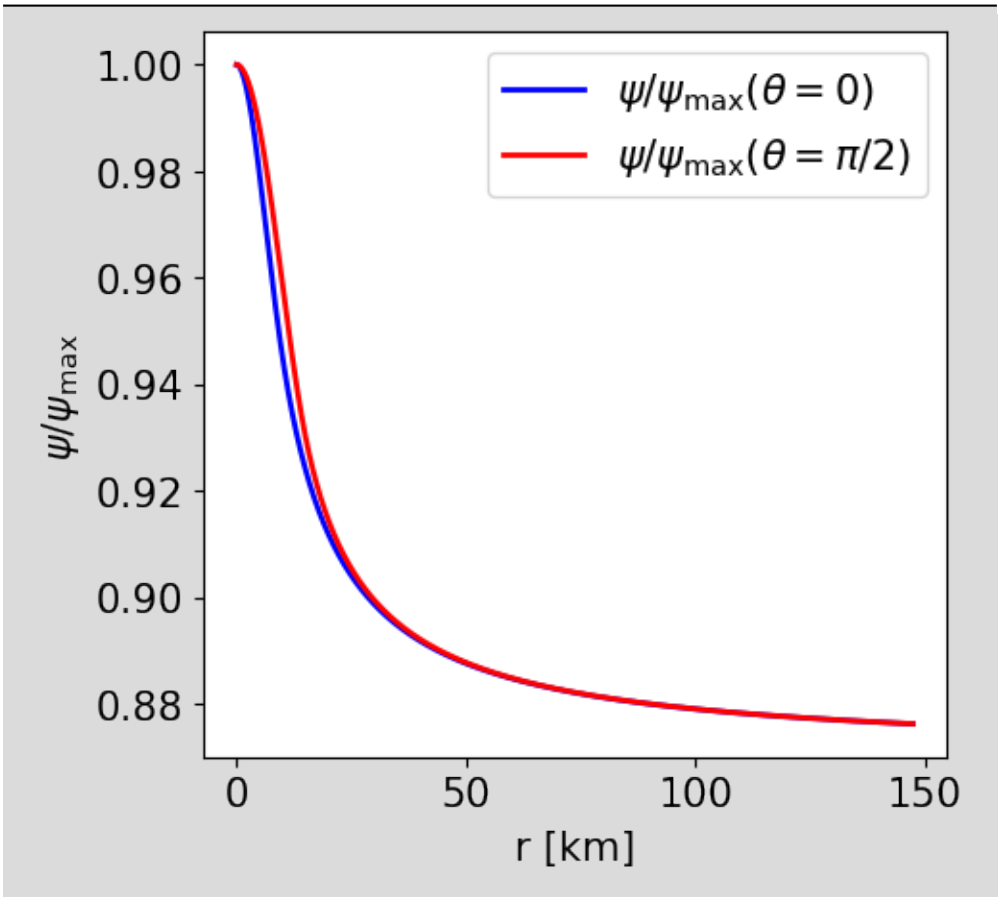
Radial profile of the NS pressure at the pole and at the equator.



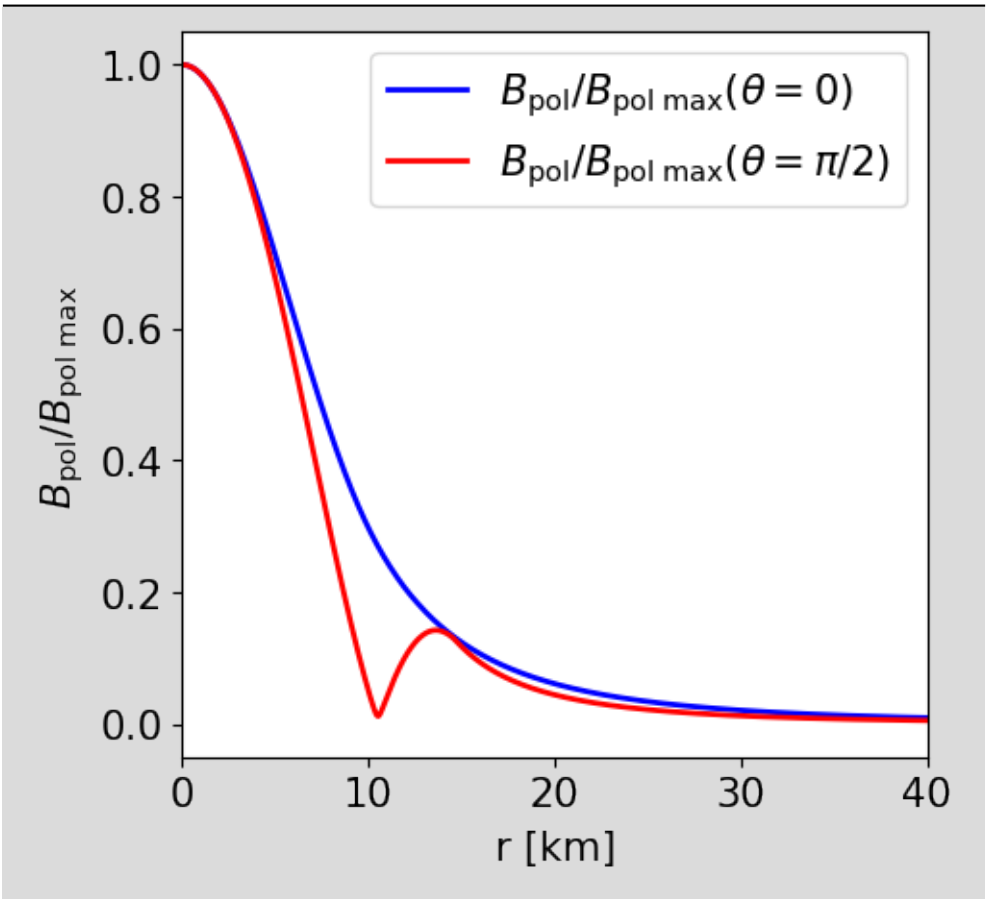
Radial profile of the scalar field at the pole and at the equator.



Radial profile of the lapse function at the pole and at the equator.



Radial profile of the conformal factor at the pole and at the equator.



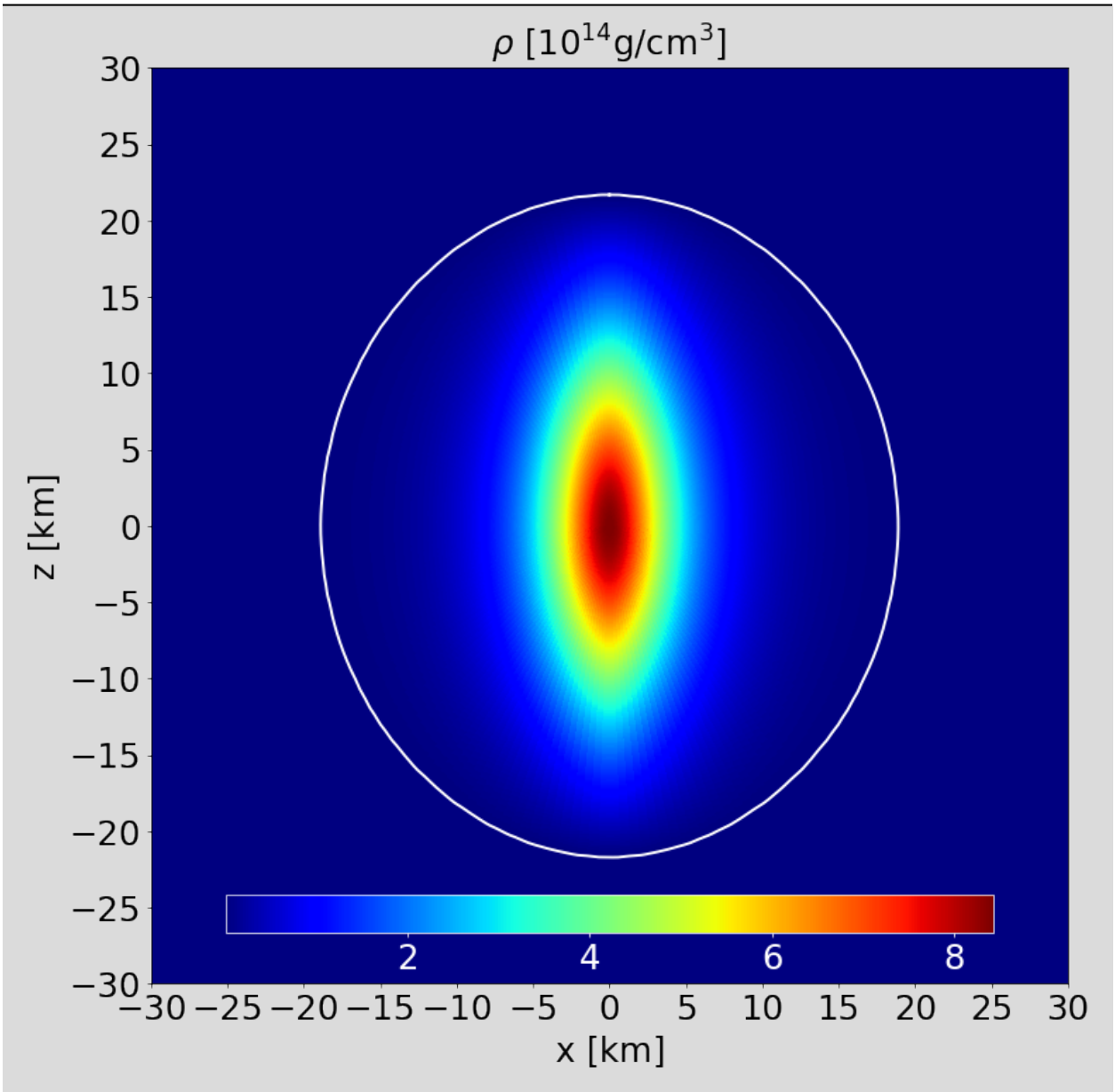
Radial profile of the poloidal magnetic field strength at the pole and at the equator.

7.6 Non-rotating NS in STT with the POL2 EoS and a purely toroidal magnetic field

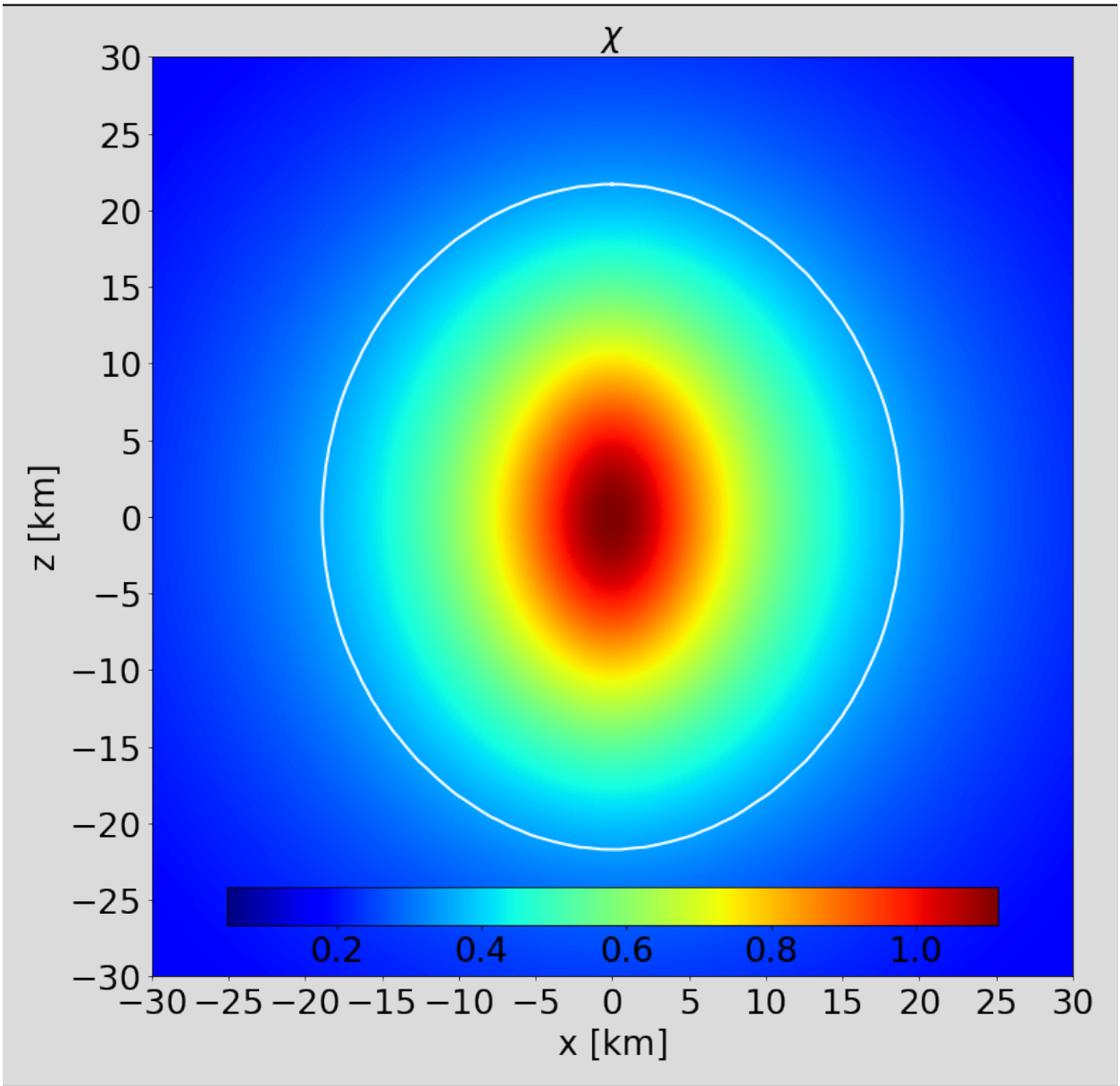
This is a model of an NS in STTs with $\alpha_0 = -2.0 \times 10^{-4}$ and $\beta_0 = -6$ described by the analytic POL2 EoS, endowed with a purely toroidal field. It has a J-frame central density $\rho_c = 1.36 \times 10^{-3}$ in code units (corresponding to 8.44×10^{14} gcm^{-3}) and a Komar mass in the E-frame of 1.467 M$_\odot$. The circumferential radius in the J-frame is 20.916 km, and the E-frame scalar charge is 0.473 M$_\odot$.

Here are the main parameters used to compute this model. The detailed log with all the parameters can be found in the corresponding LogFile.dat.

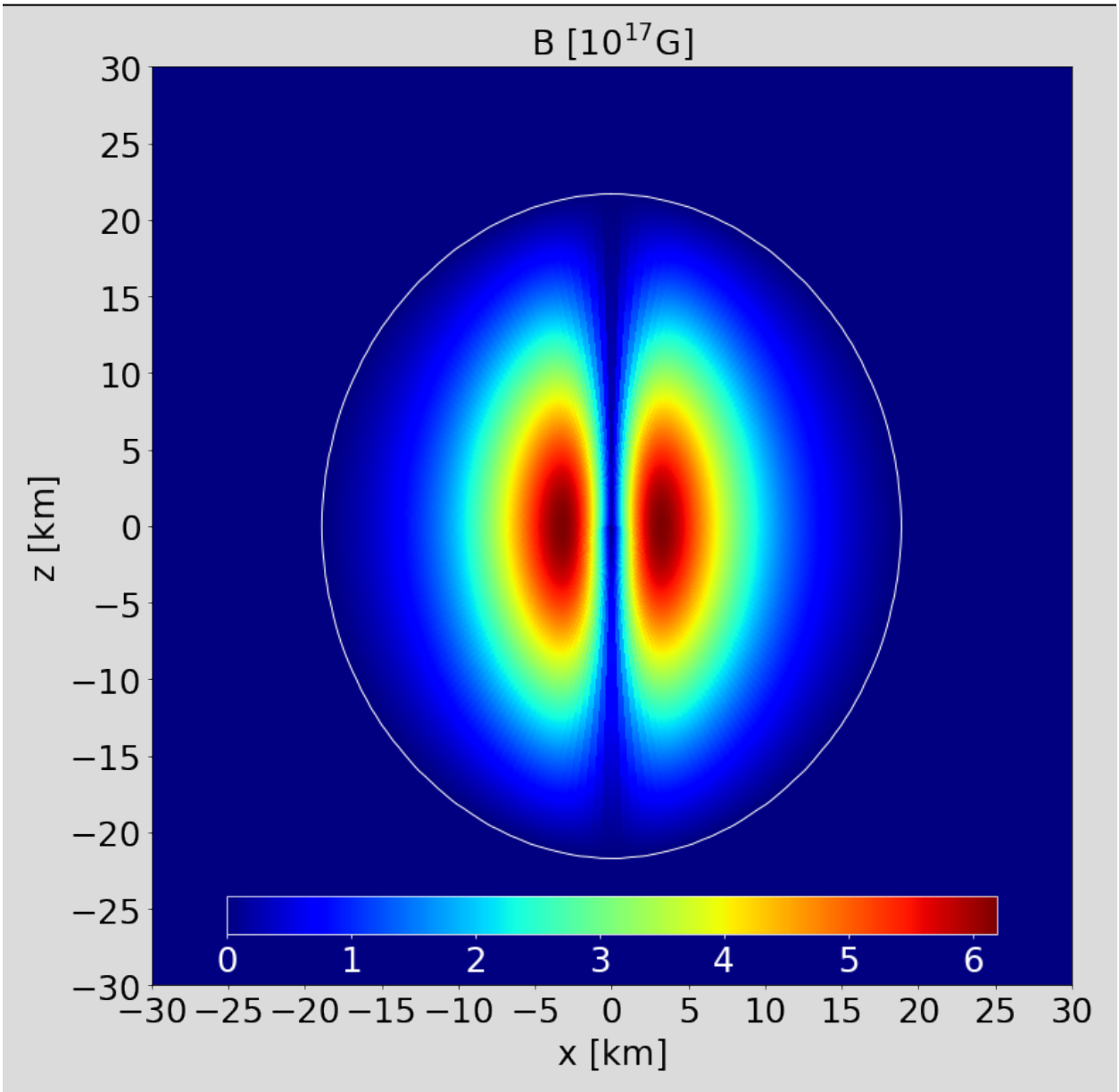
```
NR = 900, NTH = 100, NRREG = 600, MLS = 20, NGQ = 50, RREG = 20, RMAXSTR = 100,
RMAX = 100, REQMAX = 25.0, RHOINI = 1.36E-3, ALPHA0 = -2.0E-4, BETA0 = -6, GR = .FALSE.,
STRETCH = .TRUE., ANALYTIC = .TRUE., CONVHELP = .FALSE., QFACTOR = 0.85, QFACTORCHI = 0.
↪45,
QFACTORMETRIC = 0.35, QRELAX = 0.30, QAPHI = 0.50, EOSINT = .FALSE., K1 = 110.0, GAMMA =␣
↪2.0,
IMAG = .TRUE., ITOR = .TRUE., BCOEF = 4.0, NPOL = 0.0, CSI = 0.0
```



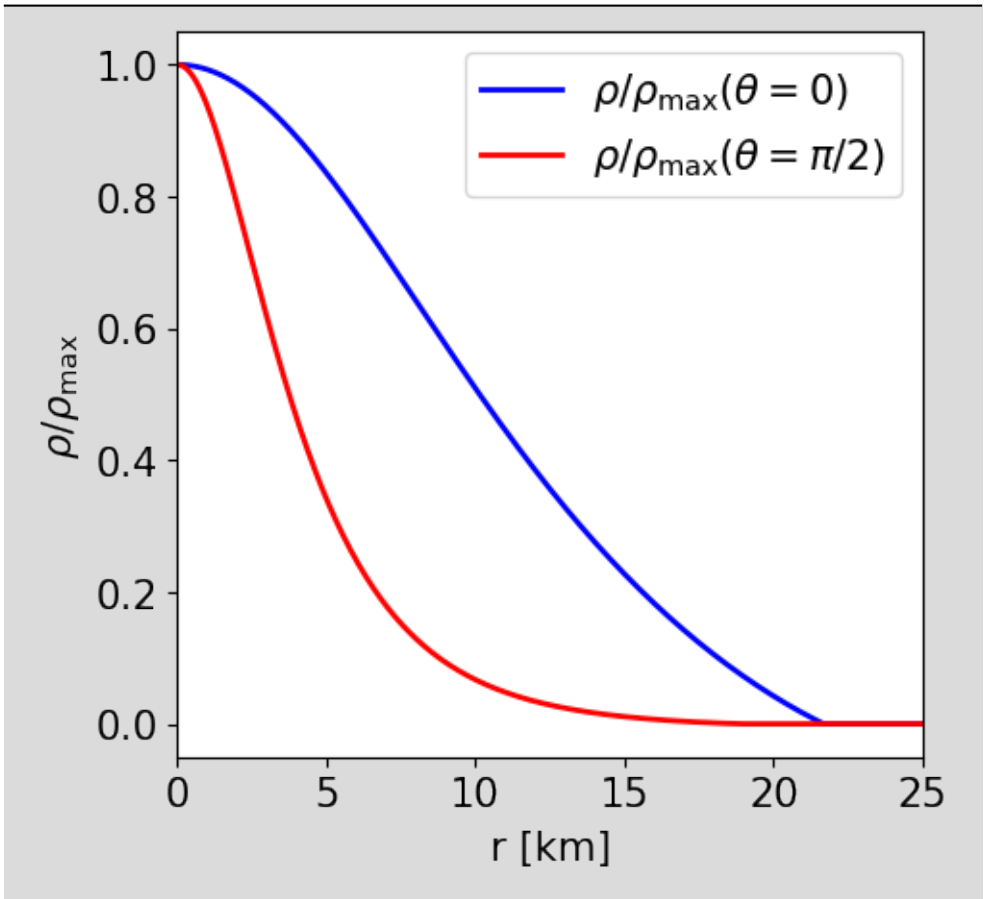
Section of the NS in the $x - z$ plane along with the contours the density.



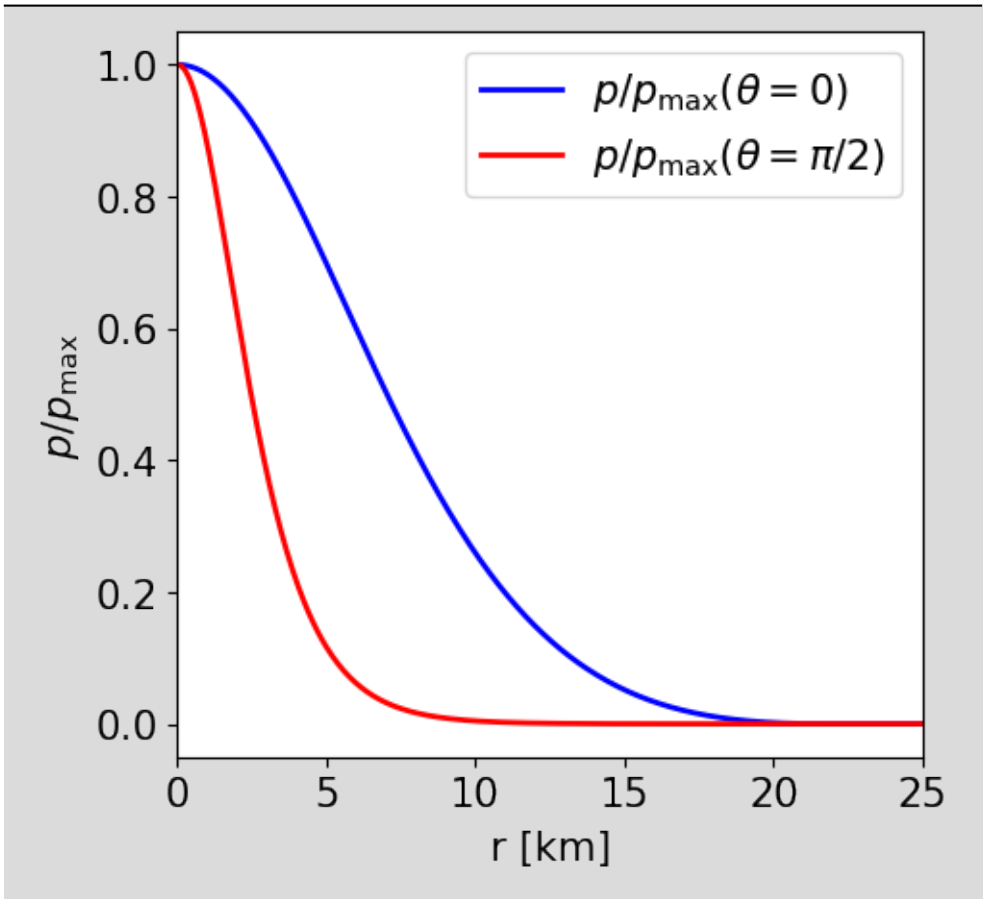
Section of the NS in the $x - z$ plane along with the contours the scalar field.



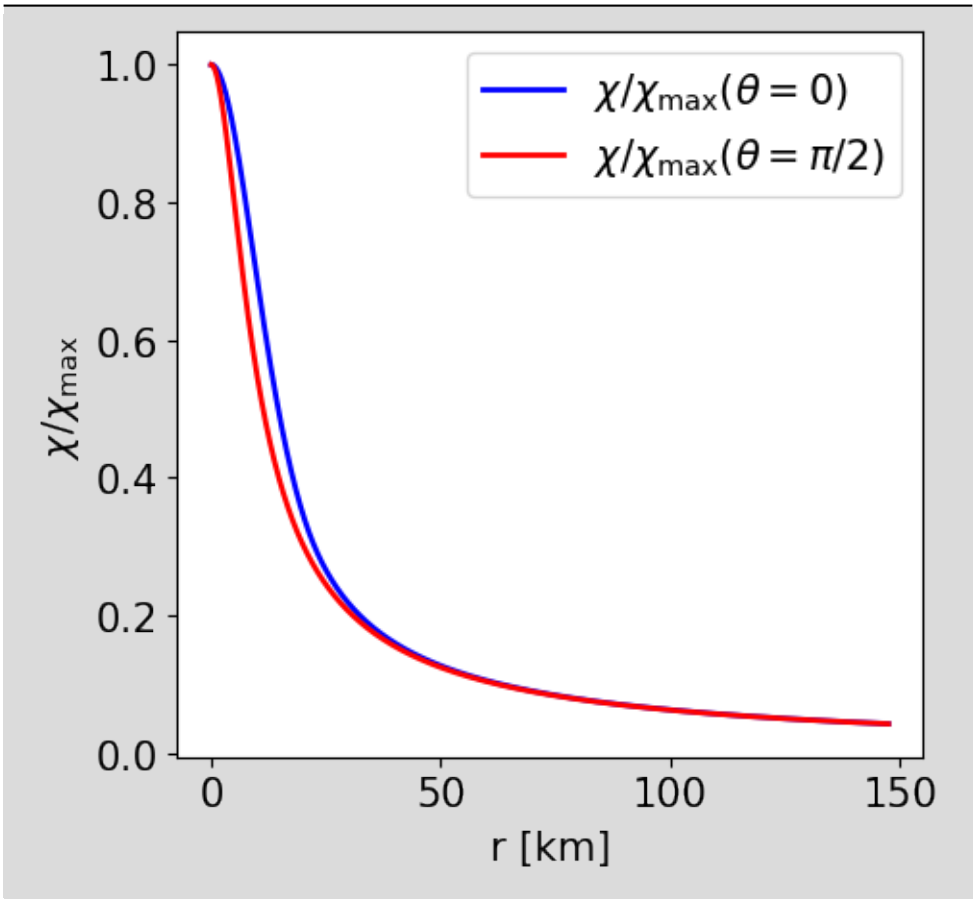
Section of the NS in the $x - z$ plane along with the contours the toroidal magnetic field strength.



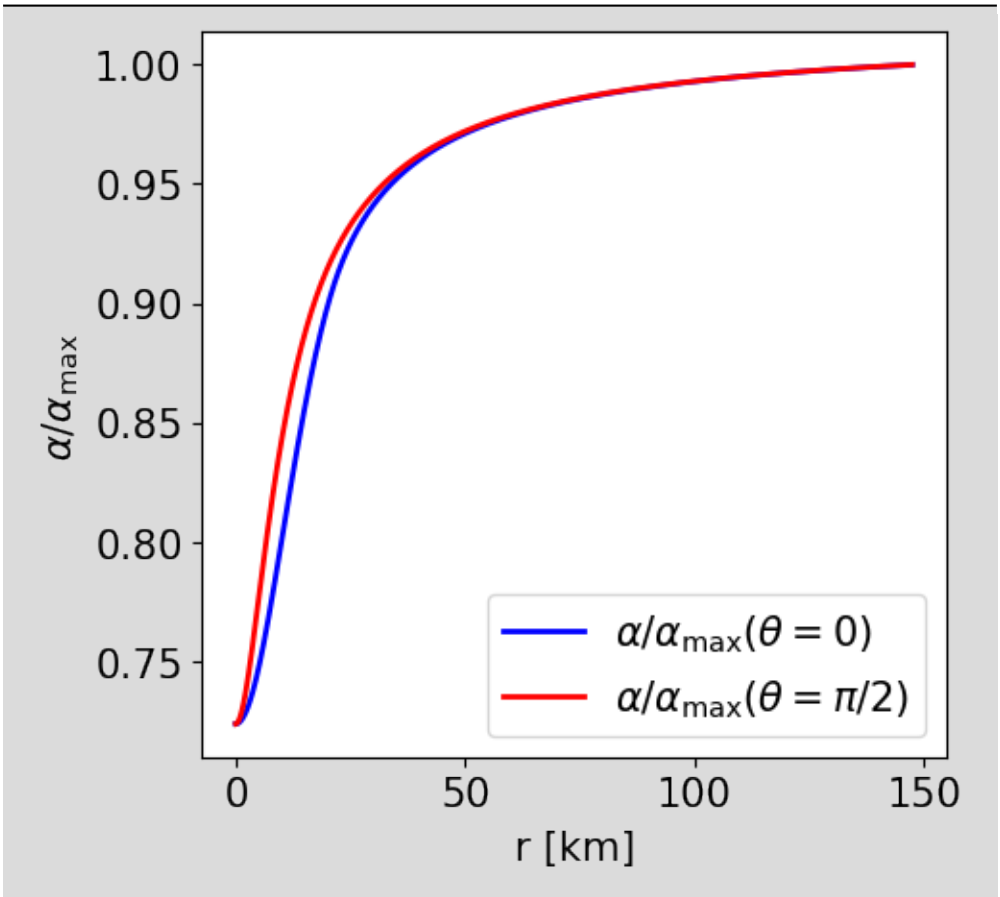
Radial profile of the NS density at the pole and at the equator.



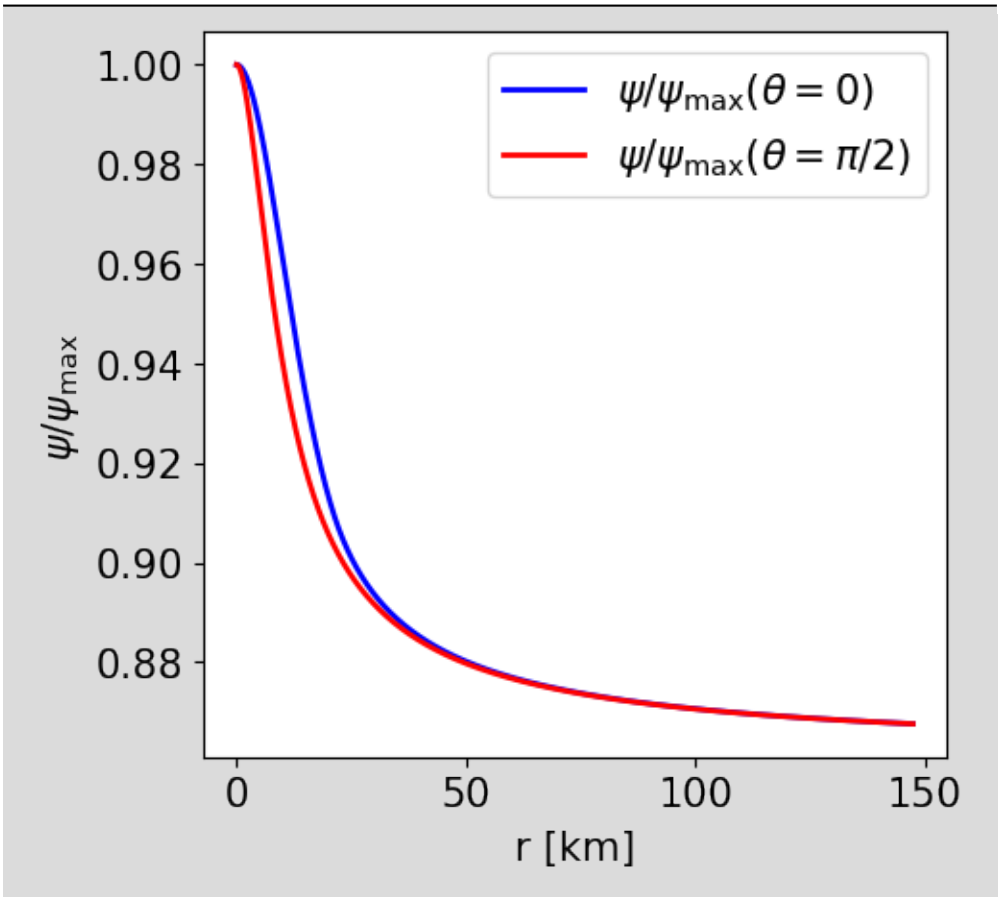
Radial profile of the NS pressure at the pole and at the equator.



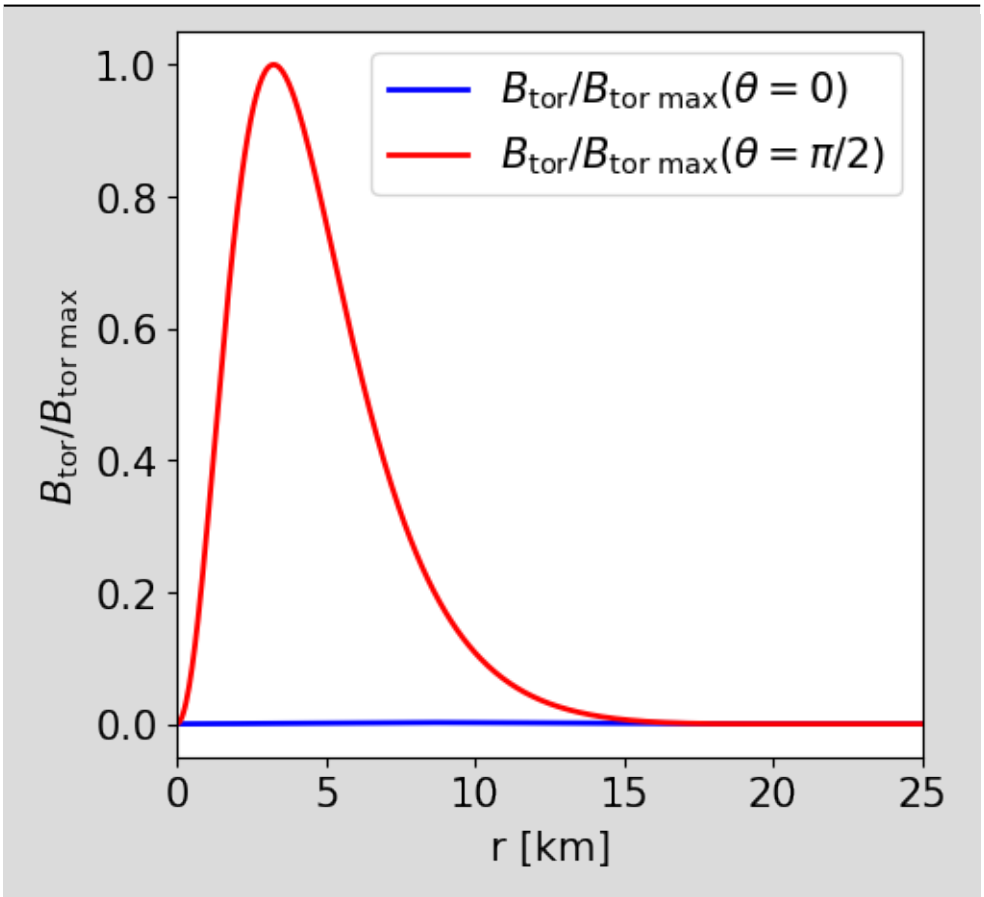
Radial profile of the scalar field at the pole and at the equator.



Radial profile of the lapse function at the pole and at the equator.



Radial profile of the conformal factor at the pole and at the equator.



Radial profile of the toroidal magnetic field strength at the pole and at the equator.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search