# Users Manual for RNS: version 2.0

Sharon Morsink

November 10, 2025

## 1 Introduction

RNS is a set of routines that will integrate the Einstein field equations for a rapidly rotating neutron star given a perfect fluid equation of state. This version of the code will be most useful for people who need to use the metric in another application. This manual will explain how to use the routines. If you have questions, please contact Sharon (morsink@ualberta.ca). If you use this code, please cite the papers listed in the Reference section.

## 2 Metric and Coordinates

The neutron star models that we compute are assumed to be stationary, axisymmetric, uniformly rotating perfect fluid solutions of the Einstein field equations. The assumptions of stationarity and axisymmetry allow the introduction of two coordinates $\phi$ and $t$ on which the space-time metric doesn't depend. The metric, $g_{\alpha\beta}$ can be written as

$$ds^2 = -e^{\gamma+\rho}dt^2 + e^{\gamma-\rho}\bar{r}^2\sin^2\theta\left(d\phi - \omega dt\right)^2 + e^{2\alpha}\left(d\bar{r}^2 + \bar{r}^2 d\theta^2\right), \tag{1}$$

where the metric potentials $\rho, \gamma, \alpha$ and $\omega$ depend only on the coordinates $\bar{r}$ and $\theta$. The function $\frac{1}{2}(\gamma+\rho)$ is the relativistic generalization of the Newtonian gravitational potential; the time dilation factor between an observer moving with angular velocity $\omega$ and an observer at infinity is $e^{\frac{1}{2}(\gamma+\rho)}$. The coordinate $\bar{r}$ is not the same as the Schwarzschild coordinate $r$. In the limit of spherical symmetry, $\bar{r}$ corresponds to the isotropic Schwarzschild coordinate. Circles centred on the axis of symmetry have circumference $2\pi r$ where $r$ is related to our coordinates $\bar{r}, \theta$ by

$$r = e^{\frac{1}{2}(\gamma-\rho)}\bar{r}\sin\theta. \tag{2}$$

The metric potential $\omega$ is the angular velocity about the symmetry axis of zero angular momentum observers (ZAMOs) and is responsible for the Lense-Thirring effect. The fourth metric potential, $\alpha$ specifies the geometry of the two-surfaces of constant $t$ and $\phi$. When the star is non-rotating, the exterior geometry is that of the isotropic Schwarzschild metric, with

$$e^{\frac{1}{2}(\gamma+\rho)} = \frac{1-M/2\bar{r}}{1+M/2\bar{r}},\ e^{\frac{1}{2}(\gamma-\rho)} = e^{\alpha} = \left(1+M/2\bar{r}\right)^2,\ \omega = 0. \tag{3}$$

The program uses a compactified coordinate $s$ which is related to $\bar{r}$ by

$$\bar{r} = \bar{r}_e\left(\frac{s}{1-s}\right), \tag{4}$$

where $\bar{r}_e$ is the value of $\bar{r}$ at the star's equator. This definition of $s$ gives

$$s = 0.5 \qquad \Leftrightarrow \bar{r} = \bar{r}_e \tag{5}$$

$$s = 1.0 \qquad \Leftrightarrow \bar{r} \to \infty \tag{6}$$

The angular variable $\mu$, defined by

$$\mu = \cos\theta$$

is used by the program.

# 3 Creating the Numerical Grid and Coordinates

The user specifies the numerical grid size in the makefile. The parameter `MDIV` (*divisions of* $\mu$) specifies the number of spokes in the angular direction, while `SDIV` (*divisions of* $s$) specifies the number of spokes in the radial direction. For example the following line in the makefile:

```
#STANDARD
SIZE=-DMDIV=65 -DSDIV=129
```

sets the number of angular spokes to 65 and the number of radial spokes to 129.

The function make_grid is called at the beginning of the application in order to set up the numerical grid.

Notes: Both MDIV and SDIV need to be odd numbers!

For most applications, larger values of MDIV and SDIV will be required. You should experiment with different grid sizes until you find that your answers converge.

## 3.1 SetUpStar()

The `rns` program starts once the user has specified the desired EOS, central density and grid size.

The first function call is: SetUpStar(). The code for this function is located in the file findmodel.c. SetUpStar() does the following tasks:

1. Read in the tabulated EOS file using load_eos()
2. Creates the coordinate grid in s and $\mu$ using make_grid
3. Allocates memory for the metric functions and their derivatives
4. Allocates memory for vectors defined on the surface of the star
5. Loads these definitions into the Star structure.

### 3.2 make_grid

The function make_grid can be found in the file: equil.c

SMAX is defined as 0.9999 in consts.h, you may want to look at the definitions of other constants, like C, G, etc in this file.

```
/*************************************************************/
/* Create computational grid.                                */
/* Points in the mu-direction are stored in the array mu[i].  */
/* Points in the s-direction are stored in the array s_gp[j]. */
/*************************************************************/
void make_grid(double s_gp[SDIV+1],
               double mu[MDIV+1])
{
  int m, s;                         /* counters */

      for(s=1;s<=SDIV;s++)
         s_gp[s] = SMAX*(s-1.0)/(SDIV-1.0);

 /* s_gp[1] = 0.0     corresponds to the center of the star
    s_gp[SDIV] = SMAX corresponds to infinity */

 /* SMAX is defined in the file consts.h */

      for(m=1;m<=MDIV;m++)
         mu[m] = (m-1.0)/(MDIV-1.0);

 /* mu[1] = 0.0    corresponds to the plane of the equator
    mu[MDIV] = 1.0 corresponds to the axis of symmetry */

 /* s_gp[0] and mu[0] are not used by the program */

}
```

### 3.3 Loading the Equation of State

RNS needs a tabulated, zero-temperature equation of state (EOS), as an input, in order to run. You will have to format the EOS file in a specific way, as described here: The first line in the EOS file should contain the number of tabulated points. The remaining lines should consist of four columns - energy density (in gr/cm$^3$), pressure (in dynes/cm$^2$), enthalpy (in cm$^2$/s$^2$), and baryon number density (in cm$^{-3}$). The *enthalpy* is defined as

$$H(P) = \int_0^P \frac{c^2 dP}{(\epsilon + P)}, \tag{7}$$

where $\epsilon$ is the energy density, $P$ is the pressure and $c$ is the speed of light.

Example files (e.g. `eosC`) are supplied in the eos repository on github:

`https://github.com/rns-alberta/eos`

# 4 Data Structures

In the `rns` code we use the following definitions of data structure types, defined in the file struct.h

At the beginning of the program a variable called "star" of type "NeutronStar" is defined. Similarly a variable called "eos" of type "EOS" is defined.

```
NeutronStar star;
EOS eos;
```

The variables star and eos are passed to most functions as pointers using the ampersand symbol.

## 4.1 `EOS`

The `EOS` structure holds information about:

1. The type of EOS (such as tabulated (default) or polytropic)
2. The name of the EOS
3. Vectors that store the values of energy density, pressure, number density, and enthalpy (in log base 10).
4. The number of points in the EOS file.

## 4.2 `Metric`

The `Metric` structure holds the following information:

1. The vector of s values at each grid point
2. The vector of $\mu$ values at each grid point
3. The matrix representing the 4 metric functions $\alpha$, $\gamma$, $\rho$, $\omega$ at each grid point
4. Matrices representing the s and $\mu$ and mixed partial derivatives of the 4 metric functions at each grid point.

## 4.3 `NeutronStar`

The `NeutronStar` structure holds the following information:

1. The Metric (defined above!)
2. Vectors defining the surface of the star (subscript surf)

3. The mass of the star
4. The equatorial radius
5. Many other physical quantities!
6. You can add anything else you like to this structure

# 5 Compute a Spherical Star

Before computing a rotating neutron star, you will need to compute the metric of a spherical neutron star. The spherical star will be used as a first approximation for the rotating neutron star.

The function MakeSphere() is called by the application to compute the metric of a spherical star. The function sphere is given information about the equation of state and the values of energy density, pressure and enthalpy at the center and surface of the star. The function computes the coordinate radius of the star $r_e$ as well as the metric functions $\rho, \gamma, \omega$ and $\alpha$.

The code for MakeSphere() can be found in findmodel.c

**Every time you want to compute a rotating star with a different central density, you must first compute the structure of a non-rotating star with the same central density using MakeSphere.**

# 6 Compute a Rotating Star

Once the spherical star's metric has been computed the application is ready to integrate the Einstein field equations for a rotating neutron star. Before doing so, the application must choose a value of the ratio of the star's polar radius to the equatorial radius. This ratio is denoted

`r_ratio`

Typically, one has to try many values of the ratio before the desired value of angular velocity, mass, etc is found.

## 6.1 Computing a rotating star given rratio

If the user has specified a desired value of rratio, then make the function call:

```
rns( r_ratio,  e_center, &eos, &star);
```

The code for rns() can be found in findmodel.c

The star structure will be loaded with most of the useful information about the star, including its mass, equatorial radius, spin frequency, and other physical quantities.

### 6.2 Computing a rotating star given spin frequency

Instead of specifying the value of rratio, you can specify a desired spin frequency in Hz (as measured by an observer at infinity). To do this make the function call

```
SetSpin(&eos, &star, e_center, spin_freq);
```

The code for SetSpin is found in findmodel.c and involves many calls to rns(). Again, the star structure will be loaded with the physical quantities computed by rns.

## 7 The Surface

The file surface.c holds routines used to find the star's surface and to compute physical quantities on the surface. To compute these quantities, make the function call:

```
Surface(&eos,&star);
```

The function Surface() does the following:

1. At each value of $\mu$, find the value of $s$ where the enthalpy vanishes.
2. Do the coordinate transformations to find the values of $\bar{r}$ and $r$ at the surface for each value of $\mu$.
3. Compute the values of each metric function and its derivatives on the surface.
4. Compute the acceleration due to gravity on the surface.
5. Store all of these quantities in the star structure.

## 8 References

Komatsu H., Eriguchi Y. & Hachisu I., 1989, *MNRAS*, **237**, 355
Cook G.B., Shapiro S.L. & Teukolsky S.A., 1994, *APJ*, **422**, 227
Stergioulas N. & Friedman J.L., 1995, *ApJ*, **444**, 306