Lorene initial data for binary neutron stars

Contents

Lorene data represents quasistationary binary neutron stars configurations, obtained by

- E. Gourgoulhon, P. Grandclément, K. Taniguchi, J.-A. Marck, S. Bonazzola, Phys. Rev. D **63**, 064029 (2001)
- K. Taniguchi, E. Gourgoulhon, S. Bonazzola, Phys. Rev. D **64**, 064012 (2001)
- K. Taniguchi, E. Gourgoulhon, Phys. Rev. D **65**, 044027 (2002)
- K. Taniguchi, E. Gourgoulhon, Phys. Rev. D **66**, 104019 (2002)
- K. Taniguchi, E. Gourgoulhon, Phys. Rev. D **68**, 124025 (2003)
- M. Bejger, D. Gondek-Rosinska, E. Gourgoulhon, P. Haensel, K. Taniguchi, J.L. Zdunik, Astron. Astrophys. **431**, 297 (2005)

The exportation of this data, computed by means of LORENE on a multi-domain spectral grid, onto a Cartesian grid (e.g. for CACTUS), is performed by means of the C++ class `Bin_NS`. The class `Bin_NS` comes along with LORENE distribution. This class is very simple, with all data members being public. A typical example of use is the following one

```
*     // Define the Cartesian grid by means of the arrays xg, yg, zg:
*     for (int i=0; i<nb_points; i++) {
*           xg[i] = ...
*           yg[i] = ...
*           zg[i] = ...
*     }
*
*     // Read the file containing the spectral data and evaluate
*     //  all the fields on the Cartesian grid :
*
*     Bin_NS binary_system(nb_points, xg, yg, zg, datafile) ;
*
*     // Extract what you need :
*
*     double* gamma_xx = binary_system.g_xx ; // metric coefficient g_xx
*
*     double* shift_x = binary_system.beta_x ; // x comp. of shift vector
*
*     ...
*
```

```
*     // Save everything in an ASCII file :
*
*     ofstream file_ini("ini.d") ;
*     binary_system.save_form(file_ini) ;
*     file_ini.close() ;
*
*
```

1

class **Bin_NS**

Binary neutron star configuration on a Cartesian grid.

Public Members

1.1	char	**eos_name1** [100]	*Eos name star 1*	10
1.2	double	**gamma_poly1**	*Adiabatic index of EOS 1 if it is polytropic (0 otherwise)*	10
1.3	double	**kappa_poly1**	*Polytropic constant of EOS 1 if it is polytropic (0 otherwise) [unit: $\rho_{\rm nuc} c^2 / n_{\rm nuc}^{\gamma}$]*	10
1.4	char	**eos_name2** [100]	*Eos name star 2*	11
1.5	double	**gamma_poly2**	*Adiabatic index of EOS 2 if it is polytropic (0 otherwise)*	11
1.6	double	**kappa_poly2**	*Polytropic constant of EOS 2 if it is polytropic (0 otherwise) [unit: $\rho_{\rm nuc} c^2 / n_{\rm nuc}^{\gamma}$]*	11
1.7	double	**omega**	*Orbital angular velocity [unit: rad/s]*	11
1.8	double	**dist**	*Distance between the centers (maxiumum density) of the two neutron stars [unit: km]*	12
1.9	double	**dist_mass**	*Distance between the center of masses of two neutron stars [unit: km]*	12
1.10	double	**mass1_b**	*Baryon mass of star 1 (less massive star) [unit: $M_\odot$]*	12
1.11	double	**mass2_b**	*Baryon mass of star 2 (massive star) [unit: $M_\odot$]*	12
1.12	double	**mass_adm**	*ADM mass of the binary system [unit: $M_\odot$]*	13
1.13	double	**angu_mom**	*Total angular momentum of the binary system [unit: $GM_\odot^2/c$]*	13

Private Members

Binary neutron star configuration on a Cartesian grid.

A binary black hole system is constructed on a Cartesian grid from data stored in a file resulting from a computation by Taniguchi and Gourgoulhon.

Importation of Lorene data is performed by means of the constructor `Bin_NS::Bin_NS(int, const double*, const double*, const double*, const char*)`. This constructor takes general arrays for the location of the Cartesian coordinates (x, y, z), i.e. it does not assume that the grid is a uniform one. Note also that these arrays are 1-D, as well as all the metric fields, in order to be use with any ordering of the 3-D storage.

This class is very simple, with all data members being public. A typical example of use is the following one

```
 *     // Define the Cartesian grid by means of the arrays xg, yg, zg:
 *     for (int i=0; i<nb_points; i++) {
 *           xg[i] = ...
 *           yg[i] = ...
 *           zg[i] = ...
 *     }
 *
 *     // Read the file containing the spectral data and evaluate
 *     //  all the fields on the Cartesian grid :
 *
 *     Bin_NS binary_system(nb_points, xg, yg, zg, datafile) ;
 *
 *     // Extract what you need :
 *
 *     double* gamma_xx = binary_system.g_xx ; // metric coefficient g_xx
 *
 *     double* shift_x = binary_system.beta_x ; // x comp. of shift vector
 *
 *     ...
 *
 *     // Save everything in an ASCII file :
 *
 *     ofstream file_ini("ini.d") ;
 *     binary_system.save_form(file_ini) ;
 *     file_ini.close() ;
 *
 *
```

Version: `$Id: bin_ns.h,v 1.5 2010/07/14 16:47:30 e_gourgoulhon Exp $`

1.1

char **eos_name1** [100]

Eos name star 1

Eos name star 1

1.2

double **gamma_poly1**

Adiabatic index of EOS 1 if it is polytropic (0 otherwise)

Adiabatic index of EOS 1 if it is polytropic (0 otherwise)

1.3

double **kappa_poly1**

Polytropic constant of EOS 1 if it is polytropic (0 otherwise) [unit: $\rho_{\rm nuc} c^2 / n_{\rm nuc}^\gamma$]

Polytropic constant of EOS 1 if it is polytropic (0 otherwise) [unit: $\rho_{\rm nuc} c^2 / n_{\rm nuc}^\gamma$]

1.4

char **eos_name2** [100]

Eos name star 2

Eos name star 2

1.5

double **gamma_poly2**

Adiabatic index of EOS 2 if it is polytropic (0 otherwise)

Adiabatic index of EOS 2 if it is polytropic (0 otherwise)

1.6

double **kappa_poly2**

Polytropic constant of EOS 2 if it is polytropic (0 otherwise) [unit: $\rho_{\rm nuc} c^2 / n_{\rm nuc}^\gamma$]

Polytropic constant of EOS 2 if it is polytropic (0 otherwise) [unit: $\rho_{\rm nuc} c^2 / n_{\rm nuc}^\gamma$]

1.7

double **omega**

Orbital angular velocity [unit: rad/s]

Orbital angular velocity [unit: rad/s]

1.8

double **dist**

Distance between the centers (maxiumum density) of the two neutron stars [unit: km]

Distance between the centers (maxiumum density) of the two neutron stars [unit: km]

1.9

double **dist_mass**

Distance between the center of masses of two neutron stars [unit: km]

Distance between the center of masses of two neutron stars [unit: km]

1.10

double **mass1_b**

Baryon mass of star 1 (less massive star) [unit: $M_\odot$]

Baryon mass of star 1 (less massive star) [unit: $M_\odot$]

1.11

double **mass2_b**

Baryon mass of star 2 (massive star) [unit: $M_\odot$]

Baryon mass of star 2 (massive star) [unit: $M_\odot$]

1.12

double **mass_adm**

ADM mass of the binary system [unit: $M_{\odot}$]

ADM mass of the binary system [unit: $M_{\odot}$]

1.13

double **angu_mom**

Total angular momentum of the binary system [unit: $GM_{\odot}^2/c$]

Total angular momentum of the binary system [unit: $GM_{\odot}^2/c$]

1.14

double **rad1_x_comp**

Coordinate radius of star 1 (less massive star) parallel to the x axis toward the companion star [unit: km]

Coordinate radius of star 1 (less massive star) parallel to the x axis toward the companion star [unit: km]

1.15

double **rad1_y**

Coordinate radius of star 1 (less massive star) parallel to the y axis [unit: km]

Coordinate radius of star 1 (less massive star) parallel to the y axis [unit: km]

1.16

double **rad1_z**

Coordinate radius of star 1 (less massive star) parallel to the z axis [unit: km]

Coordinate radius of star 1 (less massive star) parallel to the z axis [unit: km]

1.17

double **rad1_x_opp**

Coordinate radius of star 1 (less massive star) parallel to the x axis opposite to the companion star [unit: km]

Coordinate radius of star 1 (less massive star) parallel to the x axis opposite to the companion star [unit: km]

1.18

double **rad2_x_comp**

Coordinate radius of star 2 (massive star) parallel to the x axis toward the companion star [unit: km]

Coordinate radius of star 2 (massive star) parallel to the x axis toward the companion star [unit: km]

1.19

double **rad2_y**

Coordinate radius of star 2 (massive star) parallel to the y axis [unit: km]

Coordinate radius of star 2 (massive star) parallel to the y axis [unit: km]

1.20

double **rad2_z**

Coordinate radius of star 2 (massive star) parallel to the z axis [unit: km]

Coordinate radius of star 2 (massive star) parallel to the z axis [unit: km]

1.21

double **rad2_x_opp**

Coordinate radius of star 2 (massive star) parallel to the x axis opposite to the companion star [unit: km]

Coordinate radius of star 2 (massive star) parallel to the x axis opposite to the companion star [unit: km]

1.22

int **np**

Total number of grid points

Total number of grid points

1.23

double* **xx**

1-D array storing the values of coordinate x of the `np` grid points [unit: km]

1-D array storing the values of coordinate x of the `np` grid points [unit: km]

1.24

double* **yy**

1-D array storing the values of coordinate y of the `np` *grid points [unit: km]*

1-D array storing the values of coordinate y of the `np` grid points [unit: km]

1.25

double* **zz**

1-D array storing the values of coordinate z of the `np` *grid points [unit: km]*

1-D array storing the values of coordinate z of the `np` grid points [unit: km]

1.26

double* **nnn**

Lapse function N at the `np` *grid points (1-D array)*

Lapse function N at the `np` grid points (1-D array)

1.27

double* **beta_x**

Component β^x of the shift vector of non rotating coordinates [unit: c]

Component β^x of the shift vector of non rotating coordinates [unit: c]

1.28

double* **beta_y**

Component β^y of the shift vector of non rotating coordinates [unit: c]

Component β^y of the shift vector of non rotating coordinates [unit: c]

1.29

double* **beta_z**

Component β^z of the shift vector of non rotating coordinates [unit: c]

Component β^z of the shift vector of non rotating coordinates [unit: c]

1.30

double* **g_xx**

Metric coefficient γ_{xx} at the grid points (1-D array)

Metric coefficient γ_{xx} at the grid points (1-D array)

1.31

double* **g_xy**

Metric coefficient γ_{xy} at the grid points (1-D array)

Metric coefficient γ_{xy} at the grid points (1-D array)

1.32

double* **g_xz**

Metric coefficient γ_{xz} at the grid points (1-D array)

Metric coefficient γ_{xz} at the grid points (1-D array)

1.33

double* **g_yy**

Metric coefficient γ_{yy} at the grid points (1-D array)

Metric coefficient γ_{yy} at the grid points (1-D array)

1.34

double* **g_yz**

Metric coefficient γ_{yz} at the grid points (1-D array)

Metric coefficient γ_{yz} at the grid points (1-D array)

1.35

double* **g_zz**

Metric coefficient γ_{zz} at the grid points (1-D array)

Metric coefficient γ_{zz} at the grid points (1-D array)

1.36

double* **k_xx**

Component K_{xx} of the extrinsic curvature at the grid points (1-D array) [unit: c/km]

Component K_{xx} of the extrinsic curvature at the grid points (1-D array) [unit: c/km]

1.37

double* **k_xy**

Component K_{xy} of the extrinsic curvature at the grid points (1-D array) [unit: c/km]

Component K_{xy} of the extrinsic curvature at the grid points (1-D array) [unit: c/km]

1.38

double* **k_xz**

Component K_{xz} of the extrinsic curvature at the grid points (1-D array) [unit: c/km]

Component K_{xz} of the extrinsic curvature at the grid points (1-D array) [unit: c/km]

1.39

double* **k_yy**

Component K_{yy} of the extrinsic curvature at the grid points (1-D array) [unit: c/km]

Component K_{yy} of the extrinsic curvature at the grid points (1-D array) [unit: c/km]

1.40

double* **k_yz**

Component K_{yz} of the extrinsic curvature at the grid points (1-D array) [unit: c/km]

Component K_{yz} of the extrinsic curvature at the grid points (1-D array) [unit: c/km]

1.41

double* **k_zz**

Component K_{zz} of the extrinsic curvature at the grid points (1-D array) [unit: c/km]

Component K_{zz} of the extrinsic curvature at the grid points (1-D array) [unit: c/km]

1.42

double* **nbar**

Baryon density in the fluid frame at the `np` *grid points (1-D array) [unit:* $\mathrm{kg\,m^{-3}}$*]*

Baryon density in the fluid frame at the `np` grid points (1-D array) [unit: $\mathrm{kg\,m^{-3}}$]

1.43

double* **ener_spec**

Specific internal energy at the `np` *grid points (1-D array) [unit:* c^2*]*

Specific internal energy at the `np` grid points (1-D array) [unit: c^2]

1.44

double* **u_euler_x**

Component U^x *of the fluid 3-velocity with respect to the Eulerian observer, at the* `np` *grid points (1-D array) [unit:* c*]*

Component U^x of the fluid 3-velocity with respect to the Eulerian observer, at the `np` grid points (1-D array) [unit: c]

1.45

double* **u_euler_y**

Component U^y *of the fluid 3-velocity with respect to the Eulerian observer, at the* `np` *grid points (1-D array) [unit:* c*]*

Component U^y of the fluid 3-velocity with respect to the Eulerian observer, at the `np` grid points (1-D array) [unit: c]

1.46

double* **u_euler_z**

Component U^z of the fluid 3-velocity with respect to the Eulerian observer, at the `np` *grid points (1-D array) [unit: c]*

Component U^z of the fluid 3-velocity with respect to the Eulerian observer, at the `np` grid points (1-D array) [unit: c]

1.47

Bin_NS (int nbpoints, const double* xi, const double* yi, const double* zi, const char* filename)

Constructor from Lorene spectral data.

Constructor from Lorene spectral data.

This constructor takes general arrays `xi, yi, zi` for the location of the Cartesian coordinates (x, y, z), i.e. it does not assume that the grid is a uniform one. These arrays are 1-D to deal with any ordering of a 3-D storage.

Parameters:

`nbpoints` [input] Total number of grid points

`xi` [input] 1-D array (size `nbpoints`) storing thevalues of coordinate x of the grid points [unit: km]

`yi` [input] 1-D array (size `nbpoints`) storing thevalues of coordinate y of the grid points [unit: km]

`zi` [input] 1-D array (size `nbpoints`) storing thevalues of coordinate z of the grid points [unit: km]

`filename` [input] Name of the (binary) file containing the resultof a computation by means of the multi-domain spectral method.

1.48

Bin_NS (FILE*)

Constructor from a binary file (previously created by `save_bin`)

Constructor from a binary file (previously created by `save_bin`)

1.49

Bin_NS (ifstream&)

Constructor from a formatted file (previously created by `save_form`)

Constructor from a formatted file (previously created by `save_form`)

1.50

˜Bin_NS ()

Destructor

Destructor

1.52

void **save_bin** (FILE*) const

Save in a binary file.

Save in a binary file. This file can be subsenquently read by the evolution code, or by the constructor `Bin_NS::Bin_NS(FILE* )`.

1.53

void **save_form** (ofstream&) const

Save in a formatted file.

Save in a formatted file. This file can be subsenquently read by the evolution code, or by the constructor `Bin_NS::Bin_NS(ifstream& )`.

1.51

void **alloc_memory** ()

Allocate the memory for the arrays g_ij, k_ij, etc

Allocate the memory for the arrays g_ij, k_ij, etc

Class Graph