PROCRUSTES: A computer algebra package for post-Newtonian calculations in General Relativity

Dirk Puetzfeld

Institute of Theoretical Astrophysics, University of Oslo, P.O. Box 1029, 0315 Oslo, Norway

Abstract

We report on a package of routines for the computer algebra system Maple which supports the explicit determination of the geometric quantities, field equations, equations of motion, and conserved quantities of General Relativity in the post-Newtonian approximation. The package structure is modular and allows for an easy modification by the user. The set of routines can be used to verify hand calculations or to generate the input for further numerical investigations.

Key words: Approximation methods, Equations of motion, Post-Newtonian approximation, General Relativity
PACS: 04.25.Nx, 04.25.-g, 95.30.Sf

Program summary

Title of the program: Procrustes

Catalogue identifier:

Program obtainable from: CPC Program Library, author's webpage

Computer for which the program is designed and others on which it has been tested: Computers:
Platforms supported by the Maple computer algebra system (program was written

Email address: dirk.puetzfeld@astro.uio.no (Dirk Puetzfeld).
URL: www.thp.uni-koeln.de/∼dp (Dirk Puetzfeld).

under Maple 8, but also tested with Maple 9, 9.5, 10)

Operating systems under which the program has been tested:
Linux, Unix, Windows XP

Programming language used:
Maple internal language

Memory required to execute typical problem:
Dependent on problem (small $\sim$ couple of MBytes, large $\sim$ several GBytes)

Classification:
1.5 Relativity and Gravitation, 5 Computer Algebra

No. bits in a word:
Dependent on Maple distribution (supports 32 bit and 64 bit platforms)

No. of processors used: 1

No. of bytes in distributed program, including test data, etc.: $\sim$150 kbyte

Distribution format: Compressed tar file, compressed zip archive

Nature of the physical problem:
The post-Newtonian approximation represents an approximative scheme frequently used in General Relativity in which the gravitational potential is expanded into a series in inverse powers of the speed of light. Depending on the desired approximation level the field equations and equations of motion have to be determined up to given orders in the speed of light. This usually requires large algebraic computations due to the geometrical quantities entering the field equations and equations of motion.

Method of solution:
Automated computation using computer algebra techniques. Program has modular structure and only makes use of basic features of Maple to guarantee maximum compatibility and to allow for rapid extensions/modifications by the user.

Typical running time:
Dependent on problem (small $\sim$ couple of minutes, large $\sim$ couple of hours)

Restrictions on the complexity of the problem:
Sufficient amount of memory is the limiting factor for these calculations. The structure of the program allows one to handle large scale problems in an iterative manner to minimize the amount of memory required.

1 Introduction

Due to the structure of the field equations and the geometrical quantities entering these equations, computations in General Relativity are usually very tedious and prone to errors if performed by hand. This is especially true for calculations in which one makes use of some approximation scheme, such as series expansions. The so-called post-Newtonian approximation, in which certain quantities are developed in powers of the speed of light, is an example of such an expansion scheme. The post-Newtonian approximation was used in many works to determine the equations of motion in an iterative way [18,13,11], see also [19] for a more comprehensive list, and plays an important role in the experimental verification of General Relativity [21].

In contrast to the efforts in the field of exact solutions, in which computer algebra methods are heavily used, and several specialized packages [7,2,3,1,4] for common computer algebra systems [5,6,9,8] exist, there do not seem to be many works which rely on the use of such methods when working in the post-Newtonian approximation. One example of a package which has been utilized in a post-Newtonian context can be found in [17]. Further literature on the use of computer algebra methods in General Relativity as well as a list of special purpose systems is given in [15,14].

In this work we present a set of routines for the computer algebra system Maple [5]. The routines can be used to calculate the explicit form of the field equations, equations of motion, and conserved quantities in an automated fashion. The order of approximation is controlled by a set of switches and allows for a quick modification by the user. The output produced by the program can be used to support and verify hand calculations or to provide the input for subsequent numerical analysis. A splitting of time and resource intensive calculations into several small subproblems is supported by the modular structure of the routines.

The structure of the present work is as follows. In section 2 we provide a general overview of the package and a listing of different types of routines. Subsequently, we discuss a typical application in section 3, where in we also give several explicit examples on the usage of different routines in the package and discuss their input/output. In section 4 we summarize the details of our package and discuss possible future extensions and modifications. Appendix

A contains a comprehensive directory of switches and functions available in Procrustes.

2 Program structure

2.1 General thoughts

The set of routines which we summarize here under the name *Procrustes* emerged from the author's need to tackle a specific computation [19] within the post-Newtonian approximation. They were not designed to cover every imaginable problem in such a realm. The original task was to explicitly calculate the geometrical objects, field equations, and equations of motion for a given metric ansatz and to display the results in the form of a series expansion.

Since our goal was to create a program which is easy to understand and modify, we did *not* make use of any of the existing systems for tensor manipulations. Consequently, there is *no* built-in differentiation between upper and lower indices of objects, just operations with lists.

Because of the ease of use of algebraic manipulations and simplifications, we chose to implement our routines in the Maple [5] system. This program is available for many platforms and operating systems, supports large amounts of memory, has a large set of built in routines for symbolic computations/ simplifications, and supports all kinds of import and export functions. We only make use of very basic functions of Maple, especially its flexible list data structure, and intentionally use only the simplest save command, instead of the generation of a package repository which leads to a version dependence, to store our set of routines.

2.2 Routine and program structure

As mentioned in the introduction we are concerned with the post-Newtonian approximation of General Relativity. Since General Relativity represents a metric theory of gravitation, subsequent geometrical objects can be deduced from the metric. In the case of the post-Newtonian approximation the metric is developed into a series of inverse powers of the speed of light c starting from the Newtonian limit, i.e. formally we have $g_{\alpha\beta} = g^{\text{Newton}}_{\alpha\beta} + c^{-1} \overset{1}{g}_{\alpha\beta} + c^{-2} \overset{2}{g}_{\alpha\beta} + \ldots$ The strategy within the post-Newtonian approximation then typically consists of rewriting the field equations in a form which closely resembles the structures which we already know from Newtonian gravity. In addition the

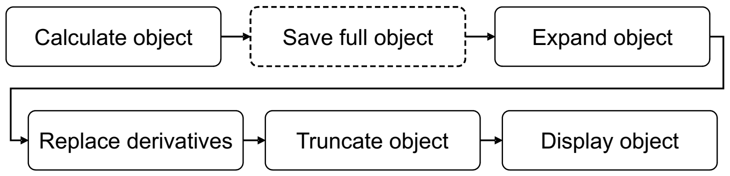

Fig. 1. Sketch of a routine which calculates and displays an object. The dashed line indicates that saving the object to a file is optional.

slow-motion condition employed in the post-Newtonian approximation leads to a hierarchy between the different metric components when one considers a split into temporal and spatial components. The assumption of slow motions also motivates the series expansion of the velocities, which enter the energy-momentum tensor, by inverse powers of the speed of light, i.e. $u^\alpha = \overset{0}{u}{}^\alpha + c^{-1} \overset{1}{u}{}^\alpha + c^{-2} \overset{2}{u}{}^\alpha + \dots$ We are not going into further details of the post-Newtonian approximation here but refer the reader to the overviews [13,16,12,20,10] and to the extensive literature list given in [19].

Our main concern, in this article, is the introduction of expansions for the metric $g_{\alpha\beta}$ and velocity u^α, in inverse powers of the speed of light, in all subsequent quantities to be calculated. In addition we must take care of the fact that derivatives with respect to the time coordinate increase the order of smallness of a term by c.

We handle the increase of the order of smallness via derivative replacements. In all routines the replacement is performed after the full object has been calculated and only plays a role in the expression which is subsequently used to produce the output (see also figure 1). The display order of the output is directly passed to the routine `calc_<object name>(<display order>)`. Only the full object remains in memory. Depending on the definition of the metric and other matter variables, which enter through energy-momentum tensor of the system under consideration, the user has to supply a list of substitution rules for all occurring derivatives with respect to the time coordinate for these variables. This substitution list has to be provided in the global variable `sublist`. We note that for the correct operation of the substitution procedure it is mandatory that `sublist` contains a hierarchical list of time derivative replacements starting with the highest expected derivative. In order to help the user with the generation of such a substitution list the package contains a small routine which produces a valid list for a given set of variable names (see the corresponding section in the sample worksheet provided with the package).

In addition to the routine which performs the actual calculation of an object, named `calc_<object name>`, there is also a routine, named `red_<object name>`, which performs a truncation of the object at the specified order. The truncation routines should be used in order to reduce the amount of memory

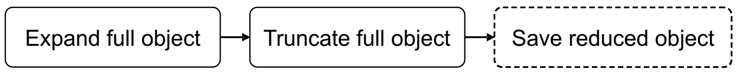

Fig. 2. Sketch of a routine which reduces an object. The dashed line indicates that saving the object to a file is optional.

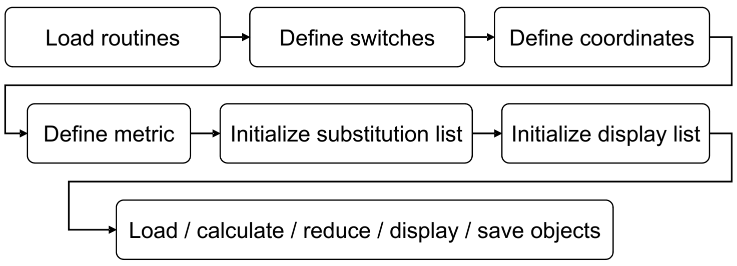

Fig. 3. Preferred structure for a calculation.

needed in the calculation of subsequent objects, such as the reduction of the connection before the curvature is calculated. The general structure of the reduction routines is sketched in figure 3. The display order of the output is directly passed to the routine `red_<object name>(<display order>)`. No derivative substitution is performed and only the truncated version of the full object remains in memory. Note that the reduction routines do *not* make use of the substitution list.

The routines in the Procrustes package are simple procedures which operate on a set of global variables; see tables A.2–A.6 for a list of routines and table A.7 for a list of variables. All routines are stored and loaded from a subdirectory defined in the global variable `CCprocrustes_dir`. We intentionally used Maple's `save` command to store the routines, instead of creating a package repository which leads to a dependence on the underlying Maple version. All of the routines can be found in the file `procrustes_<version number>.mws`, the user may modify this file to replace existing or add his/her own routines. A simple execution of this worksheet will store the routines in the package subdirectory.

The recommendation for the general structure of a worksheet to be used with the package is sketched in figure 3. In the next section we provide some detailed comments on the input and output of the demonstration worksheet `procrustes_demo.mws` supplied together with the package.

3 Typical application

In this section we comment on a sample calculation with Procrustes. The input/output displayed here can be produced with the worksheet `procrustes_demo.mws` supplied with the package. We note that this worksheet allows one to quickly reproduce many of the results of the classic paper [11].

3.1 Installation

To install the package create a directory of your choice and, depending on the operating system, unpack the `tar` or `zip` archive. The directory should contain two files `procrustes_<version number>.mws` (containing the source of the package), and `procrustes_demo.mws` (a demonstration worksheet; excerpts of this file are discussed in the sections below). The subdirectories `procrustes<version number>` (containing the set of routines generated from the source file), and `testrun_procrustes<version number>` (containing data from a testrun of the package, which is used by the worksheet to determine if the package is installed correctly).

Modifications of the package can be performed by editing the source of the routines in `procrustes_<version number>.mws`. When introducing new routines we recommend maintaining the same naming convention as introduced in the original file (see also the complete list of routines in tables A.2-A.6). New routines should be added to the save list at the end of the file in order to guarantee their storage for further usage. A simple execution of the worksheet then stores the modified/new routines in the package subdirectory.

In an actual calculation we recommend to load the required set of routines from the package subdirectory. The worksheet `procrustes_demo.mws` contains, apart from some instructive sections on how to specify metrics and substitution lists, a sample calculation which sticks to the scheme sketched in figure 3.

3.2 Coordinate and metric specification

The coordinates to be used by Procrustes have to be specified in form of a list in a global variable named `xx`, in the first section of the worksheet this is done by the command: [1]

[1] In the following input by the user is marked by "`>`" and output by Maple is preceeded by "`<`". Three vertically aligned dots are used to indicated that we shortened the displayed input/output.

```
> xx:=[x0,x1,x2,x3];
```

Based on these coordinates the metric has to be specified in form of a list. The worksheet contains a section in which the contravariant as well as the covariant form of the metric are specified, the latter one `gdd` being defined as:

```
> gdd:=[[1-2*U(xx[1],xx[2],xx[3],xx[4])/c^2+phi(xx[1],xx[2],xx[3],xx[4])/c^4,hdd01(xx[1],xx[2],xx[3],xx[4]/c^3,
         hdd02(xx[1],xx[2],xx[3],xx[4]/c^3,hdd03(xx[1],xx[2],xx[3],xx[4])/c^3],
        [hdd01(xx[1],xx[2],xx[3],xx[4])/c^3,(-1-2*V(xx[1],xx[2],xx[3],xx[4])/c^2),0,0],
        [hdd02(xx[1],xx[2],xx[3],xx[4])/c^3,0,(-1-2*V(xx[1],xx[2],xx[3],xx[4])/c^2),0],
        [hdd03(xx[1],xx[2],xx[3],xx[4])/c^3,0,0,(-1-2*V(xx[1],xx[2],xx[3],xx[4])/c^2)]];
```

This definition corresponds to the line element

$$
\begin{aligned}
ds^2 &= g_{\alpha\beta} dx^\alpha dx^\beta \\
&= \left(1 - \frac{2U}{c^2} + \frac{\phi}{c^4}\right) dx^0 dx^0 + \frac{2h_{0a}}{c^3} dx^0 dx^a - \left(1 + \frac{2V}{c^2}\right) \delta_{ab} dx^a dx^b, \qquad (1)
\end{aligned}
$$

where U, h_{0a}, and V represent free functions which depend on the coordinates indicated. We remind the reader that there is no built-in distinction between upper and lower indices, hence also the contravariant form `guu` of the metric has to be provided by hand (see sample worksheet). Coordinate as well as metric definitions are then saved to the files `coordinates_demo.mpi, gdd_demo.mpi, guu_demo.mpi` for later usage in the sample calculation.

3.3 Substitution and display list specification

As we already have mentioned in section 2 the increase in the order of smallness of quantities is handled via derivative replacements. The substitution list has to be provided in a global variable named `sublist` and should contain a hierarchical list of replacement rules, starting with highest expected derivative of a variable. In the sample worksheet we have included a section in which a substitution list is explicitly specified. This list is subsequently used in the sample calculation and has the following structure:

```
> sublist:=[diff(U(x0,x1,x2,x3),x0,x0,x0,x0)=diff(U(t,x1,x2,x3),t,t,t,t)/c^4,
            diff(U(x0,x1,x2,x3),x0,x0,x0)=diff(U(t,x1,x2,x3),t,t,t)/c^3,
            diff(U(x0,x1,x2,x3),x0,x0)=diff(U(t,x1,x2,x3),t,t)/c^2,
            diff(U(x0,x1,x2,x3),x0)=diff(U(t,x1,x2,x3),t)/c,
            U(x0,x1,x2,x3)=U(t,x1,x2,x3),
            .
            .
            .
```

We have added a procedure called `generate_sublist` to assists the user in the generation of a correct substitution list for all variables in a calculation.

Apart from the substitution list, the package also makes use of a display list, to be defined in the global variable `displist`. The purpose of this list is to bring the output from the routines in the package into a neat form which can be quickly interpreted by the user. The display list does *not* play a role in the actual calculation of quantities, nor in the storage of partial results, it is only used when a procedure outputs an object on the screen (see figure 1). In the sample worksheet provided the display list is used to suppress the functional dependencies of the variables in the calculation, thereby reducing the length of the output considerably. The example for `displist` given therein has the following structure:

```
> displist:=[U(t,x1,x2,x3)=U,
            V(t,x1,x2,x3)=V,
            rho(t,x1,x2,x3)=rho,
            pp(t,x1,x2,x3)=p,
            .
            .
            .
```

Depending on the needs of the user one could also use the display list to pre-process output which can then be directly exported into LaTeX. Either by exclusively using the display list or by a combination of the display list and Maple's built-in output facility. In the next section we show some input and output examples from the worksheet. We caution the user that quantities which were reduced by the substitutions in the display list should *not* be used for any further calculations.

3.4 Calculation and reduction of objects

After the specification of the coordinates, metric, substitution list, and display list we are ready to perform a calculation with the package. In the demonstration worksheet we provide a sample for the general structure of a worksheet for calculations with Procrustes. After a preamble in which we load the desired procedures from the package directory, and the specification of some global switches (see the list in table A.1), we make use of the previous definitions of the metric and the corresponding substitution list by loading them from the previously specified files and then perform some actual calculations. For example, the command

```
> calc_uu(4);
```

calculates and displays the contravariant four velocity up to third order in inverse powers of the speed of light c,

```
<                            2           2       2
                         vu03        vu01    vu02
                         ----- + U + ----- + -----
                           2           2       2
 Velocity up[, 0, ], 1 + -------------------------
                                      2
                                     c


                                3            2                     2
                            vu01    vu01 vu03             vu01 vu02
                            ----- + ---------- + vu01 U + ----------
                     vu01     2         2                     2
 Velocity up[, 1, ], ---- + ----------------------------------------
                      c                        3
                                              c
  .
  .
  .
```

This corresponds to the result

$$u^0 = 1 + \frac{1}{c^2}\left(\frac{1}{2}\delta_{ab}v^a v^b + U\right), \quad u^1 = \left[\frac{1}{c} + \frac{\frac{1}{2}\delta_{ab}v^a v^b + U}{c^3}\right] v^1, \quad \ldots \qquad (2)$$

In addition to the actual calculation of objects it is sometimes advantageous to reduce the order of an object, i.e. to remove higher orders in its series expansion which are not needed in subsequent calculations. Such a reduction is supported by functions starting with `red_<object name>`, which exist for most of the objects calculable with the package, cf. table A.5. In contrast to the routines which calculate objects and display them up to a given order, the reduction routines truncate the expanded version of the full object before derivative substitutions are performed, see also the scheme sketched in figure 3. As an example we calculate the connection from the sample metric,

```
> calc_condndndn(4);

<                                             d
                                              -- U
                                              dt
             Connection (condndndn)[, 0, 0, 0, ], - ----
                                                3
                                               c
 .
 .
 .
```

Note that the order passed to the calculation command only influences up to the order objects are displayed, *not* the actual form of the object which is kept in memory for further calculations. We can display the full object by directly accessing the corresponding list element, which in this case is:

```
> condndndn[1,1,1];

<
            d                        d
           --- U(x0, x1, x2, x3)    --- phi(x0, x1, x2, x3)
           dx0                      dx0
         - --------------------- + 1/2 -----------------------
                   2                        4
                  c                        c
```

Hence, we see that the full object is still in memory. Only the reduce command truncates the full object and thereby affects subsequent calculations, as we can see from the following command sequence:

```
> red_condndndn(4);condndndn[1,1,1];

<
                         d
                        --- U(x0, x1, x2, x3)
                        dx0
                      - ---------------------
                                2
                               c
```

This also shows that no derivative replacements are performed during the reduction of full objects. The maximum order up to which objects get truncated is controlled via the global variable `CCmaxorder` (cf. table A.1). The user should make sure that the order specified in this variable is high enough to guarantee that all unwanted higher order terms are removed. If the user wants to display a single component of an object including the derivative replacements the command `disp_component`, cf. table A.2, should be used. For the connection component from the previous examples this command yields

```
> disp_component('condndndn[1,1,1]',4);

<
                                        d
                                        -- U
                                        dt
```

```
                        condndndn[1, 1, 1], - ----
                                               3
                                              c
```

Finally, we note that the reduction commands should be used with great care, since they directly influence the calculation of subsequent objects if these depend on the object being reduced.

3.5 Saving and loading components

Time and memory intensive calculations are supported by the package via an integrated save function, which is controlled by the global switch `CCoutputfs` (see table A.1). If this switch is set to the value 2, all functions produce a file with reusable output. The calculation routines produce output files with names `<object/component name>.full.mpi`. Files with the names `<object/component name>.red.mpi` are generated by the reduction routines. Output files are generated for every component of an indexed object and there are routines for loading the entire object as well as only single components. The sample worksheet contains a section in which the previously computed results are loaded from files. For all objects there exists a corresponding load routine `load_<object name>(full/reduced switch)` which reads the complete object from files, the switch passed to the load command determines whether files with the suffix `.full.mpi` or `.red.mpi` are used (see table A.6 for a complete list).

Components of objects can be loaded from the existing files by using the `load_component` command. In contrast to the `load_<object name>` command the `load_component` command returns the specified component of an object but does *not* automatically assign it to the corresponding variable name. Here are some examples for objects with one and two indices and a scalar quantity:

```
> load_component('uu',[1],1); load_component('tupup',[1,2],2); load_component('thetascalar',[],2);

<                          "uu_1.full.mpi"
                           "tupup_1_2.red.mpi"
                           "thetascalar.red.mpi"
```

As noted before the above commands do not automatically perform assignments to the corresponding variable names. This can be easily done by hand as follows

```
> for aa from 1 to 4 do test_uu[aa]:=load_component('uu',[aa],2) od:

<                           "uu_1.red.mpi"
                            "uu_2.red.mpi"
                            "uu_3.red.mpi"
                            "uu_4.red.mpi"
```

or by using the predefined load command, which loads all components of `uu` from the corresponding files and assigns them. We can verify the equivalence of the last command sequence and the load command by

```
> load_uu(2);

<                           "uu_1.red.mpi"
```

```
                          "uu_2.red.mpi"
                          "uu_3.red.mpi"
                          "uu_4.red.mpi"

> seq(is(test_uu[aa]=uu[aa]),aa=1..4);

<                     true, true, true, true
```

More examples can be found in the sample worksheet. The complete list of predefined load commands is given in table A.6.

3.6 Testrun

After the installation it is advisable to perform a quick testrun with the package. This can easily be done by executing the worksheet supplied with the package. After performing the sample calculations (which should not take longer than a couple of minutes on a modern desktop machine [2]) the worksheet automatically compares its results with some test data files which are supplied together with the package (as noted before the data for comparison is located in the directory `testrun_procrustes<version number>`). All of the current features of the package are demonstrated in the sample worksheet. We note that the calculation in the sample worksheet reproduces many results of the classic work [11] and is therefore a good example for the effectiveness of the package.

4 Conclusions and outlook

We have introduced the main features of the Procrustes package and explained how it can be used to tackle computations within the post-Newtonian approximation of General Relativity. The package is versatile and should allow the user to easily make modifications. Possible future developments of the package strongly depend on the scientific problems in which the author will be involved. Planned future developments include: (i) An automated order control for all quantities entering a specific post-Newtonian calculation. This feature would further reduce the memory requirements of the program and would alleviate the user from making choices for the correct post-Newtonian orders of single quantities. (ii) Automated simplification schemes for the output. The goal is to introduce some extra routines which match certain patterns in the output and thereby reduce the need for manual calculations even further. (iii) Extension of the package to incorporate non-Riemannian theories of gravitation. The complexity of the geometric quantities in such theories renders virtually

[2] With Maple 8 it takes about 35 seconds on a Pentium M (1.7 GHz) system and approximately 320 MBytes of main memory.

Table A.1
List of global switches.

Name	Description
`CCmaxorder`	Order up to which expanded objects get truncated. The standard setting is `200`, meaning that the reduction of objects is performed starting from the order passed to the routine up to c^{-200}.
`CCoutputfs`	Switch which toggles the output to files. The standard setting is `2`, meaning that all routines will produce files with their output.
`CCprocrustes_dir`	Directory in which the routines of the package are stored, the standard setting is `"./procrustes<versionumber>/"`.
`CCtaylor`	Order variable used in Taylor approximations, the standard setting is `3`.

any hand calculation unfeasible. From a physical point of view this is one of the most pressing steps, since it would allow us to establish a systematic post-Newtonian framework for alternative gravitational theories.

Acknowledgements

The author is grateful to M. Pohl (Iowa State University) for his longterm support of this project. In addition the support by Ø. Elgarøy (University of Oslo) and the Research Council of Norway (project number 162830) during the later stages of this work is gratefully acknowledged. For the supply and support of different Maple versions the author would like to thank J. Veverka (Iowa State University), T. Leifsen (University of Oslo), and U. Fuskeland (University of Oslo). Finally, the author is grateful to two anonymous referees whose constructive comments helped to improve the manuscript.

A Directory of functions and names

In this section we provide the names and a description of predefined switches A.1, a list of routines and their dependencies A.2-A.6, and a list of global variables A.7 in table form.

Table A.2
List of general predefined functions.

Name	Description
`system_status()`	Outputs memory usage of the current session
`fapprox(<expression>,<order>,<point>)`	Taylor approximation of the inverse square root up to the given order at a given point
`gapprox(<expression>,<order>,<point>)`	Taylor approximation of the square root up to the given order at a given point
`disp_component('<object name>',<display order>)`	Displays an object in memory up to the given order
`load_component(<object name>,<index>,<full/reduced expression>)`	Loads object components from file, last switch used to toggle between full ("=1") / reduced ("=2") expressions

References

[1] Cartan distribution. http://www.adinfinitum.no/cartan/.

[2] EinS distribution. http://rcswww.urz.tu-dresden.de/~klioner/eins.html.

[3] Excalc documentation. http://www.uni-koeln.de/REDUCE/3.6/doc/excalc.

[4] GRTensor distribution. http://grtensor.phy.queensu.ca/.

[5] Maple distribution. http://www.maplesoft.com/.

[6] Mathematica distribution. http://www.wolfram.com/.

[7] MathTensor distribution. http://smc.vnet.net/MathTensor.html.

[8] MuPad distribution. http://www.mupad.de/.

[9] Reduce distribution. http://www.reduce-algebra.com/.

[10] V.A. Brumberg. *Essential relativistic celestial mechanics.* Adam Hilger, Bristol, 1991.

[11] S. Chandrasekhar. The post-Newtonian equations of hydrodynamics in General Relativity. *Astrophys. J.*, 142:1488, 1965.

[12] T. Damour. The problem of motion in Newtonian and Einsteinian gravity. *300 years of gravitation, Cambridge Univ. Press, Eds. Hawking & Israel*, page 128, 1987.

Table A.3
List of predefined functions which calculate an object and display it up to the given `<display order>`.

Name	Dependencies	Description
`calc_uu(<display order>)`	`gdd, fapprox()`	Contravariant four velocity $u^\alpha = dx^\alpha/ds$
`calc_ud(<display order>)`	`gdd, uu`	Covariant four velocity $u_\alpha = g_{\alpha\beta}u^\beta$
`calc_tupup(<display order>)`	`uu, guu`	Contravariant perfect fluid energy-momentum tensor $T^{\alpha\beta}$
`calc_tdndn(<display order>)`	`ud, gdd`	Covariant perfect fluid energy-momentum tensor $T_{\alpha\beta} = \left(\rho c^2 + \Pi\rho + p\right) u_\alpha u_\beta - p g_{\alpha\beta}$
`calc_tupdn(<display order>)`	`gdd, tupup`	Mixed perfect fluid energy-momentum tensor $T^\alpha{}_\beta$
`calc_rhs(<display order>)`	`gdd, tdndn, tupdn`	Right-hand side of the field equations $\mathrm{RHS}_{\alpha\beta} = -\frac{\kappa}{c^4}\left(T_{\alpha\beta} - \frac{1}{2}T^\gamma{}_\gamma g_{\alpha\beta}\right)$
`calc_condndndn(<display order>)`	`gdd`	Connection $\Gamma_{\gamma\|\alpha\beta} = \frac{1}{2}\left(g_{\gamma\alpha,\beta} + g_{\beta\gamma,\alpha} - g_{\alpha\beta,\gamma}\right)$
`calc_conupdndn(<display order>)`	`guu, condndndn`	Connection $\Gamma^\gamma_{\alpha\beta} = g^{\gamma\delta}\Gamma_{\delta\|\alpha\beta}$
`calc_ctupup(<display order>)`	`tupup, conupdndn`	Covariant derivative of the EM tensor $T^{\alpha\beta}{}_{;\gamma}$
`calc_eom(<display order>)`	`ctupup`	Equations of motion $T^{\alpha\beta}{}_{;\beta} = 0$
`calc_ricdndn(<display order>)`	`guu, gdd, condndndn`	Covariant Ricci tensor $R_{\mu\nu} = \frac{1}{2}g^{\alpha\beta}(g_{\mu\nu,\alpha\beta} + g_{\alpha\beta,\mu\nu} - g_{\alpha\nu,\mu\beta} - g_{\mu\beta,\alpha\nu}) + g^{\alpha\beta}g^{\gamma\delta}(\Gamma_{\delta\|\mu\nu}\Gamma_{\gamma\|\alpha\beta} - \Gamma_{\gamma\|\mu\beta}\Gamma_{\delta\|\alpha\nu})$
`calc_ricupdn(<display order>)`	`guu, ricdndn`	Mixed Ricci tensor $R^\alpha{}_\beta$
`calc_ricscalar(<display order>)`	`ricupdn`	Mixed Ricci scalar $R = R^\alpha{}_\alpha$
`calc_riemupdndndn(<display order>)`	`conupdndn`	Riemann curvature tensor $R^\alpha{}_{\beta\mu\nu} = -\Gamma^\alpha_{\beta\mu,\nu} + \Gamma^\alpha_{\beta\nu,\mu} - \Gamma^\gamma_{\beta\mu}\Gamma^\alpha_{\gamma\nu} + \Gamma^\gamma_{\beta\nu}\Gamma^\alpha_{\gamma\mu}$
`calc_riemdndndndn(<display order>)`	`gdd, riemupdndn`	Riemann curvature tensor $R_{\alpha\beta\mu\nu} = g_{\alpha\gamma}R^\gamma{}_{\beta\mu\nu}$
`calc_weyldndndndn(<display order>)`	`gdd, ricdndn, ricscalar, riemdndndndn`	Weyl curvature tensor $C_{\alpha\beta\mu\nu} = R_{\alpha\beta\mu\nu} - \frac{1}{2}(g_{\alpha\nu}B_{\beta\mu} + g_{\beta\mu}B_{\alpha\nu} - g_{\alpha\mu}B_{\beta\nu} - g_{\beta\nu}B_{\alpha\mu}) - \frac{1}{12}R(g_{\alpha\nu}g_{\beta\mu} - g_{\alpha\mu}g_{\beta\nu})$ with $B_{\alpha\beta} = R_{\alpha\beta} - \frac{1}{4}g_{\alpha\beta}R$
`calc_detguu(<display order>)`	`guu`	Metric determinant $det(g^{\alpha\beta})$
`calc_detgdd(<display order>)`	`gdd`	Metric determinant $det(g_{\alpha\beta})$

Table A.4
List of predefined functions which calculate an object and display it up to the given `<display order>`.

Name	Dependencies	Description
`calc_pemtupup(<display order>)`	`guu, conupdndn`	Landau-Lifshitz energy-momentum pseudotensor $t^{\mu\nu} = \frac{c^4}{2\kappa}[(2\Gamma^{\delta}_{\alpha\beta}\Gamma^{\kappa}_{\delta\kappa} - \Gamma^{\delta}_{\alpha\kappa}\Gamma^{\kappa}_{\beta\delta} - \Gamma^{\delta}_{\alpha\delta}\Gamma^{\kappa}_{\beta\kappa})(g^{\mu\alpha}g^{\nu\beta} - g^{\mu\nu}g^{\alpha\beta}) + g^{\mu\alpha}g^{\beta\delta}(\Gamma^{\nu}_{\alpha\kappa}\Gamma^{\kappa}_{\beta\delta} + \Gamma^{\nu}_{\beta\delta}\Gamma^{\kappa}_{\alpha\kappa} - \Gamma^{\nu}_{\delta\kappa}\Gamma^{\kappa}_{\alpha\beta} - \Gamma^{\nu}_{\alpha\beta}\Gamma^{\kappa}_{\delta\kappa}) + g^{\nu\alpha}g^{\beta\delta}(\Gamma^{\mu}_{\alpha\kappa}\Gamma^{\kappa}_{\beta\delta} + \Gamma^{\mu}_{\beta\delta}\Gamma^{\kappa}_{\alpha\kappa} - \Gamma^{\mu}_{\delta\kappa}\Gamma^{\kappa}_{\alpha\beta} - \Gamma^{\mu}_{\alpha\beta}\Gamma^{\kappa}_{\delta\kappa}) + g^{\alpha\beta}g^{\delta\kappa}(\Gamma^{\mu}_{\alpha\delta}\Gamma^{\nu}_{\beta\kappa} - \Gamma^{\mu}_{\alpha\beta}\Gamma^{\nu}_{\delta\kappa})]$
`calc_thetaupup(<display order>)`	`detgdd, tupup, pemtupup`	Landau-Lifshitz energy-momentum complex $\Theta^{\alpha\beta} = -\det(g_{\alpha\beta})[T^{\alpha\beta} + t^{\alpha\beta}]$
`calc_thetaupuppdn(<display order>)`	`thetaupup`	Partial derivative of the LL EM complex ${\Theta^{\alpha\beta}}_{,\beta}$
`calc_harm_gc(<display order>)`	`guu, detgdd, gapprox()`	Harmonic gauge condition $(\sqrt{-g}g^{\alpha\beta})_{,\beta}$
`calc_harm_gc_alt(<display order>)`	`guu, conupdndn`	Harmonic gauge condition $g^{\alpha\beta}\Gamma^{\gamma}_{\alpha\beta}$ (alternative form)
`calc_pn_gc(<display order>)`	`gdd`	Standard post-Newtonian gauge condition $(\delta^{ab}(g_{0a,b} - \frac{1}{2}g_{ab,0}) = 0,\ \delta^{bc}g_{ab,c} - \frac{1}{2}(\delta^{bc}g_{bc} - g_{00})_{,a} = 0)$
`calc_cuu(<display order>)`	`uu, conupdndn`	Covariant derivative of the four velocity ${u^{\alpha}}_{;\beta}$
`calc_cud(<display order>)`	`ud, conupdndn`	Covariant derivative of the four velocity $u_{\alpha;\beta}$
`calc_thetascalar(<display order>)`	`cuu`	Expansion scalar $\theta = {u^{\alpha}}_{;\alpha}$
`calc_acd(<display order>)`	`uu, cud`	Acceleration $a_{\alpha} = u_{\alpha;\beta}u^{\beta}$
`calc_omegadd(<display order>)`	`ud, cud, acd`	Rotation $\omega_{\alpha\beta} = u_{[\alpha;\beta]} + u_{[\alpha}a_{\beta]}$
`calc_thetadd(<display order>)`	`ud, cud, acd`	Symmetric part of the derivative projection $\theta_{\alpha\beta} = u_{(\alpha;\beta)} - u_{(\alpha}a_{\beta)}$
`calc_sigmadd(<display order>)`	`gdd, ud, thetadd, thetascalar`	Shear $\sigma_{\alpha\beta} = \theta_{\alpha\beta} - \frac{1}{3}\theta(g_{\alpha\beta} - u_{\alpha}u_{\beta})$
`check_velnorm(<display order>)`	`uu, ud`	Checks fulfillment of the velocity normalization
`check_metricid(<display order>)`	`guu, gdd`	Checks fulfillment of the metric identity
`check_decomposition(<display order>)`	`omegadd, sigmadd, thetascalar, gdd, ud, acd, cud`	Checks whether decomposition $u_{\alpha;\beta} = \omega_{\alpha\beta} + \sigma_{\alpha\beta} + \frac{1}{3}(g_{\alpha\beta} - u_{\alpha}u_{\beta})\theta + a_{\alpha}u_{\beta}$ is consistent

Table A.5
List of predefined functions which truncate an object at the given `<truncation order>`.

Name	Dependencies	Object
`red_uu(<truncation order>)`	`uu`	Four velocity u^α
`red_ud(<truncation order>)`	`ud`	Four velocity u_α
`red_tupup(<truncation order>)`	`tupup`	EM tensor $T^{\alpha\beta}$
`red_tdndn(<truncation order>)`	`tdndn`	EM tensor $T_{\alpha\beta}$
`red_tupdn(<truncation order>)`	`tupdn`	EM tensor $T^\alpha{}_\beta$
`red_rhs(<truncation order>)`	`rhsfeqsdndn`	Right-hand side of FEQs $\mathrm{RHS}_{\alpha\beta}$
`red_condndndn(<truncation order>)`	`condndndn`	Connection $\Gamma_{\gamma\vert\alpha\beta}$
`red_conupdndn(<truncation order>)`	`conupdndn`	Connection $\Gamma^\gamma_{\alpha\beta}$
`red_ctupup(<truncation order>)`	`ctupup`	Covariant deriv. of the EM tensor $T^{\alpha\beta}{}_{;\gamma}$
`red_eom(<truncation order>)`	`eom`	EOMs $T^{\alpha\beta}{}_{;\beta} = 0$
`red_ricdndn(<truncation order>)`	`ricdndn`	Covariant Ricci tensor $R_{\alpha\beta}$
`red_ricupdn(<truncation order>)`	`ricupdn`	Mixed Ricci tensor $R^\alpha{}_\beta$
`red_ricscalar(<truncation order>)`	`ricscalar`	Ricci scalar $R^\alpha{}_\alpha$
`red_riemupdndndn(<truncation order>)`	`riemupdndndn`	Riemann curvature tensor $R^\alpha{}_{\beta\mu\nu}$
`red_riemdndndndn(<truncation order>)`	`riemdndndndn`	Riemann curvature tensor $R_{\alpha\beta\mu\nu}$
`red_weyldndndndn(<truncation order>)`	`weyldndndndn`	Weyl curvature tensor $C_{\alpha\beta\mu\nu}$
`red_detguu(<truncation order>)`	`detguu`	Metric determinant $det(g^{\alpha\beta})$
`red_detgdd(<truncation order>)`	`detgdd`	Metric determinant $det(g_{\alpha\beta})$
`red_pemtupup(<truncation order>)`	`pemtupup`	LL EMPT $t^{\alpha\beta}$
`red_thetaupup(<truncation order>)`	`thetaupup`	LL EM complex $\Theta^{\alpha\beta}$
`red_thetaupuppdn(<truncation order>)`	`thetaupuppdn`	Partial deriv. of the LL EM complex $\Theta^{\alpha\beta}{}_{,\beta}$
`red_cuu(<truncation order>)`	`cuu`	Covariant derivative $u^\alpha{}_{;\beta}$
`red_cud(<truncation order>)`	`cud`	Covariant derivative $u_{\alpha;\beta}$
`red_thetascalar(<truncation order>)`	`thetascalar`	Expansion scalar θ
`red_acd(<truncation order>)`	`acd`	Acceleration a_α
`red_omegadd(<truncation order>)`	`omegadd`	Rotation $\omega_{\alpha\beta}$
`red_thetadd(<truncation order>)`	`thetadd`	Symm. part of the deriv. projection $\theta_{\alpha\beta}$
`red_sigmadd(<truncation order>)`	`sigmadd`	Shear $\sigma_{\alpha\beta}$

Table A.6
List of predefined functions which load objects from files. The parameter `<full/reduced expression>` passed to the function determines whether the full or reduced object is loaded (`1 = <object name>.full.mpi, 2 = <object name>.red.mpi`). All functions depend on the `load_component()` routine.

Name	Object
`load_uu(<full/reduced expression>)`	Four velocity u^{α}
`load_ud(<full/reduced expression>)`	Four velocity u_{α}
`load_tupup(<full/reduced expression>)`	EM tensor $T^{\alpha\beta}$
`load_tdndn(<full/reduced expression>)`	EM tensor $T_{\alpha\beta}$
`load_tupdn(<full/reduced expression>)`	EM tensor $T^{\alpha}{}_{\beta}$
`load_rhs(<full/reduced expression>)`	Right-hand side of FEQs $\mathrm{RHS}_{\alpha\beta}$
`load_condndndn(<full/reduced expression>)`	Connection $\Gamma_{\gamma\|\alpha\beta}$
`load_conupdndn(<full/reduced expression>)`	Connection $\Gamma^{\gamma}_{\alpha\beta}$
`load_ctupup(<full/reduced expression>)`	Covariant derivative of the EM tensor $T^{\alpha\beta}{}_{;\gamma}$
`load_eom(<full/reduced expression>)`	EOMs $T^{\alpha\beta}{}_{;\beta} = 0$
`load_ricdndn(<full/reduced expression>)`	Covariant Ricci tensor $R_{\alpha\beta}$
`load_ricupdn(<full/reduced expression>)`	Mixed Ricci tensor $R^{\alpha}{}_{\beta}$
`load_ricscalar(<full/reduced expression>)`	Ricci scalar $R^{\alpha}{}_{\beta}$
`load_riemupdndndn(<full/reduced expression>)`	Riemann curvature tensor $R^{\alpha}{}_{\beta\mu\nu}$
`load_riemdndndndn(<full/reduced expression>)`	Riemann curvature tensor $R_{\alpha\beta\mu\nu}$
`load_weyldndndndn(<full/reduced expression>)`	Weyl curvature tensor $C_{\alpha\beta\mu\nu}$
`load_detguu(<full/reduced expression>)`	Metric determinant $det(g^{\alpha\beta})$
`load_detgdd(<full/reduced expression>)`	Metric determinant $det(g_{\alpha\beta})$
`load_pemtupup(<full/reduced expression>)`	LL EMPT $t^{\alpha\beta}$
`load_thetaupup(<full/reduced expression>)`	LL EM complex $\Theta^{\alpha\beta}$
`load_thetaupuppdn(<full/reduced expression>)`	Partial derivative of the LL EM complex $\Theta^{\alpha\beta}{}_{,\beta}$
`load_cuu(<full/reduced expression>)`	Covariant velocity derivative $u^{\alpha}{}_{;\beta}$
`load_cud(<full/reduced expression>)`	Covariant velocity derivative $u_{\alpha;\beta}$
`load_thetascalar(<full/reduced expression>)`	Expansion scalar θ
`load_acd(<full/reduced expression>)`	Acceleration a_{α}
`load_omegadd(<full/reduced expression>)`	Rotation $\omega_{\alpha\beta}$
`load_thetadd(<full/reduced expression>)`	Symmetric part of the derivative projection $\theta_{\alpha\beta}$
`load_sigmadd(<full/reduced expression>)`	Shear $\sigma_{\alpha\beta}$

Table A.7
List of global variables

Name	Description
`acd`	Kinematic acceleration a_α
`condndndn, conupdndn`	Connection $\Gamma_{\gamma\|\alpha\beta}$, $\Gamma^\alpha_{\beta\gamma}$
`ctupup`	Covariant derivative of the EM tensor $T^{\alpha\beta}{}_{;\gamma}$
`cuu, cud`	Covariant derivative of the four velocity $u^\alpha{}_{;\beta}$, $u_{\alpha;\beta}$
`detgdd, detguu`	Metric determinant $det(g_{\alpha\beta})$, $det(g^{\alpha\beta})$
`displist`	Substitution list for replacements in output
`eom`	Equations of motion $T^{\alpha\beta}{}_{;\beta}$
`gdd, guu`	Covariant $g_{\alpha\beta}$ / contravariant $g^{\alpha\beta}$ metric
`harm_gc, harm_gc_alt`	Harmonic gauge condition
`omegadd`	Kinematic rotation $\omega_{\alpha\beta}$
`pemtupup`	Landau-Lifshitz EM pseudotensor $t^{\alpha\beta}$
`pn_gc`	Standard post-Newtonian gauge condition
`ricdndn, ricupdn`	Ricci tensor $R_{\alpha\beta}$, $R^\alpha{}_\beta$
`ricscalar`	Ricci scalar R
`riemupdndndn, riemdndndndn`	Riemann curvature tensor $R^\alpha{}_{\beta\gamma\delta}$, $R_{\alpha\beta\gamma\delta}$
`rhsfeqsdndn`	Right hand side of the field equations $-\frac{\kappa}{c^4}\left(T_{\alpha\beta} - \frac{1}{2}T^\gamma{}_\gamma g_{\alpha\beta}\right)$
`routine_list`	List containing the names of routines to be loaded from routine directory
`sigmadd`	Kinematic shear $\sigma_{\alpha\beta}$
`sublist`	Substitution list for derivative replacements
`tdndn, tupup, tupdn`	Energy-momentum tensor $T_{\alpha\beta}$, $T^{\alpha\beta}$, $T^\alpha{}_\beta$
`thetadd`	Kinematic quantity $\theta_{\alpha\beta}$, symmetric part of the velocity derivative projection
`thetaupup, thetaupuppdn`	Landau-Lifshitz EM complex $\Theta^{\alpha\beta}$ and its partial derivative $\Theta^{\alpha\beta}{}_{,\beta}$
`thetascalar`	Kinematic expansion θ
`ud, uu`	Covariant u_α / contravariant u^α four velocity
`velnorm`	Velocity product $u^\alpha u_\alpha$
`weyldndndndn`	Weyl curvature tensor $C_{\alpha\beta\gamma\delta}$
`xx`	List with coordinates

[13] V.A. Fock. *The theory of space time and gravitation.* Pergamon Press, New York (Orig. 1955), 1959.

[14] J. Grabmeier, E. Kaltofen, and V. Weispfenning (Eds.). *Computer Algebra Handbook.* Springer, Berlin, 2003.

[15] F.W. Hehl, R. Puntigam, and H. Ruder (Eds.). *Relativity and scientific computing.* Springer, Berlin, 1996.

[16] L. Infeld and J. Plebanski. *Motion and Relativity.* Pergamon Press, New York, 1960.

[17] S. A. Klioner. New system for indical computation and its applications in gravitational physics. *Comp. Phys. Comm.*, 115:231, 1998.

[18] H.A. Lorentz and J. Droste. The motion of a system of bodies under the influence of their mutual attraction, according to Einstein's theory. *Versl. K. Akad. Wet. Amsterdam*, 26:392, 1917.

[19] D. Puetzfeld. The cosmological post-Newtonian equations of hydrodynamics in General Relativity. *Submitted*, 2006.

[20] M.H. Soffel. *Relativity in astronomy, celestial mechanics, and geodesy.* Springer, Berlin, 1989.

[21] C.M. Will. *Theory and experiment in gravitational physics.* Cambridge University Press, Cambridge, 1993.