Eur. Phys. J. C (2021) 81:504
https://doi.org/10.1140/epjc/s10052-021-09183-9

THE EUROPEAN
PHYSICAL JOURNAL C

Special Article - Tools for Experiment and Theory

xPPN: an implementation of the parametrized post-Newtonian formalism using *xAct* for Mathematica

Manuel Hohmann[a]

Laboratory of Theoretical Physics, Institute of Physics, University of Tartu, W. Ostwaldi 1, 50411 Tartu, Estonia

Received: 3 February 2021 / Accepted: 26 April 2021

Abstract We present a package for the computer algebra system Mathematica, which implements the parametrized post-Newtonian (PPN) formalism. This package, named *xPPN*, is built upon the widely used tensor algebra package suite *xAct*, and in particular the package *xTensor* therein. The main feature of *xPPN* is to provide functions to perform a proper 3+1 decomposition of tensors, as well as a perturbative expansion in so-called velocity orders, which are central tasks in the PPN formalism. Further, *xPPN* implements various rules for quantities appearing in the PPN formalism, which aid in perturbatively solving the field equations of any metric theory of gravity. Besides Riemannian geometry, also teleparallel and symmetric teleparallel geometry are implemented.

1 Introduction

The parametrized post-Newtonian (PPN) formalism [1–7] is an indispensable tool for testing the viability of gravity theories. It is used to characterize any given theory of gravity by a set of ten parameters, which are closely related to the phenomenological properties of the theory under consideration. This allows to compare the parameters which are obtained theoretical through the PPN formalism with their values measured in experiments within the solar system and related physical settings.

In order for the PPN formalism to be applicable, the gravity theory under consideration must satisfy a number of assumptions. The most important is the existence of a metric governing the motion of test bodies, which can be described by a perturbation around a flat Minkowski background. Another assumption concerns the source of gravity, which is chosen to be a fluid satisfying the covariant conservation of energy-momentum, encoded in the Euler equations of fluid dynamics. Further, it is assumed that the field equations are of second derivative order, or can be brought into this form, so that their solution take a known standard form in terms of particular Poisson-like integrals over the source matter distribution.

In order to determine the post-Newtonian limit of a given gravity theory, one must perform a 3+1 split of its field equations (which are, in general, tensor equations) into space and time components, and then perform a perturbative expansion around a fixed vacuum solution. Depending on the structure of the field equations, both tasks may be challenging to perform by hand, and so the use of computer algebra comes as a useful tool. Due to the common assumptions on which the PPN formalism is based, and the similar steps to be applied to different theories of gravity, it appears natural to implement these common tasks into a general computational tool, which can then be used to calculate the post-Newtonian limit for any given theory. A number of functions to achieve this has already been implemented in Maple [8]. The aim of *xPPN*[9] is to provide another implementation, based on a more extensive framework for performing tensor calculations.

A very powerful tensor algebra software is the *xTensor* package, which is part of the *xAct* suite of Mathematica packages [10]. It comes with numerous functions to define and manipulate tensorial expressions, and includes concepts such as induced metrics on hypersurfaces orthogonal to a vector field, or component calculations in *xCoba*, which can be used to achieve a 3 + 1 split of tensorial expressions. The former is employed, for example, for cosmological perturbation theory in the *xPand* package [11], in conjunction with the *xPert* package [12] for general tensor perturbations. It thus appears natural to follow a similar approach to implement also the PPN formalism in *xAct*. However, the latter comes with a few peculiarities which obstruct this strategy. The main difficulty lies in the different treatment of space and time in the PPN formalism, such as assigning different perturbation orders to space and time components of tensors. Another,

[a] e-mail: manuel.hohmann@ut.ee (corresponding author)

Published online: 07 June 2021

albeit smaller issue is the different convention for counting the perturbation orders in *xPert* and the PPN formalism.

Even though it is possible to implement the PPN formalism also using the existing functionality mentioned above, it appears simpler to overcome the aforementioned difficulties by using a different approach both to the 3 + 1 split and the perturbative expansion, without using the induced metric framework or the *xCoba* and *xPert* packages. This is the approach followed by *xPPN*. The key idea is to complement every tensor defined on the spacetime manifold by a number of tensors on a purely spatial manifold, which additionally depend on an external time parameter, and which represent the separated time and space components of the original spacetime tensor. Also derivatives are split into spatial derivatives and derivatives with respect to the time parameter. This approach allows an essentially different treatment of perturbations for space and time components. The aim of this article is to present this approach, as well as the implementation of the PPN formalism which is based on this framework.

This article is structured as follows. We start with a brief review of the PPN formalism and its extensions implemented by *xPPN* in Sect. 2. The general concepts on which this implementation is based are explained in Sect. 3. The main functionality of *xPPN* is laid out in Sect. 4, where we display the geometric objects defined by *xPPN*, and Sect. 5, where we explain the functions provided for the common operations on tensor expressions. A complete usage example is given in Sect. 6, which shows how to calculate the PPN parameters of a simple scalar-tensor theory of gravity. A summary and outlook towards possible future extensions are given in Sect. 7. For further reference, we provide notes on the implementation and internal operation of *xPPN* in Appendix A.

In order to make code listings more readable, the following syntax highlighting is used. Mathematica keywords are typeset in blue, *xAct* keywords are typeset in green and *xPPN* keywords are typeset in red. We use lowercase letters for coordinate indices and uppercase letters for Lorentz indices, where in both cases Greek indices run from 0 to 3 and belong to spacetime, while Latin indices run from 1 to 3 and belong to space only.

2 Parametrized post-Newtonian formalism

The aim of the *xPPN* package is to provide an implementation of the parametrized post-Newtonian (PPN) formalism and several of its geometric extensions. Here we briefly summarize these theoretical foundations. We first discuss the standard PPN formalism and the perturbative expansion of the metric in Sect. 2.1. An extended formulation using a tetrad as the fundamental field instead of the metric is shown in Sect. 2.2. We then display its application to teleparallel gravity in Sect. 2.3 and symmetric teleparallel gravity in Sect. 2.4.

2.1 Standard formalism for Riemannian geometry

There exist different versions of the PPN formalism. Here we adhere to its form presented in [5]. Its starting point is the assumption that the propagation of light and massive test bodies is governed by a pseudo-Riemannian metric $g_{\alpha\beta}$ of Lorentzian signature. This metric is further considered in a weak field limit as an asympotically flat perturbation around a flat Minkowski background. The source of this perturbation is assumed to be of compact support and modeled by the energy-momentum tensor

$$\Theta^{\alpha\beta} = (\rho + \rho\Pi + p)u^\alpha u^\beta + pg^{\alpha\beta} \tag{1}$$

of a perfect fluid with four-velocity u^α, rest energy density ρ, specific internal energy Π and pressure p. Further, a fixed Cartesian coordinate system $(x^\alpha) = (t, x^a)$ is used, denoted as the *universe rest frame*. It is then assumed that the velocity $v^a = u^a/u^0$ of the source matter in this coordinate system is small compared to the speed of light, $|v^a| \ll c \equiv 1$. This velocity takes the role of the perturbation parameter. Physical quantities, such as the metric, are expanded in terms of the source velocity, and any term in this perturbative expansion which is proportional to $|v^a|^n$ is said to be of n'th velocity order, which is commonly denoted $\mathcal{O}(n)$ in the literature.[1] For the metric $g_{\alpha\beta}$, this expansion may be written explicitly in the form

$$g_{\alpha\beta} = \sum_{n=0}^{\infty} \overset{n}{g}_{\alpha\beta} = \overset{0}{g}_{\alpha\beta} + \overset{1}{g}_{\alpha\beta} + \overset{2}{g}_{\alpha\beta} + \overset{3}{g}_{\alpha\beta} + \overset{4}{g}_{\alpha\beta} + \mathcal{O}(5), \tag{2}$$

where we used overscripts to indicate velocity orders $\overset{n}{g}_{\alpha\beta} \sim \mathcal{O}(n)$. Terms beyond the fourth velocity order are usually not considered in the standard PPN formalism implemented by *xPPN*. The zeroth velocity order is assumed to be the flat Minkowski background,

$$\overset{0}{g}_{\alpha\beta} = \eta_{\alpha\beta} = \mathrm{diag}(-1, 1, 1, 1). \tag{3}$$

For the metric perturbations, one finds that the first velocity order vanishes, $\overset{1}{g}_{\alpha\beta} = 0$, since the lowest velocity order of the matter source is the second order, as we will argue below. Further, the components $\overset{2}{g}_{0a}$, $\overset{3}{g}_{00}$, $\overset{3}{g}_{ab}$, $\overset{4}{g}_{0a}$ change their sign under time reversal, and are prohibited by energy-momentum conservation. It follows that only the components

$$\overset{2}{g}_{00}, \quad \overset{2}{g}_{ab}, \quad \overset{3}{g}_{0a}, \quad \overset{4}{g}_{00}, \quad \overset{4}{g}_{ab} \tag{4}$$

are non-vanishing. For the first four terms, a particular standard gauge is assumed, in which the metric components take

[1] Note in particular that $\mathcal{O}(1) \sim |v|$ in this notation is the first velocity order, and not unity, which would be $\mathcal{O}(0) \sim 1$.

the form

$$\overset{2}{g}_{00} = 2U, \tag{5a}$$

$$\overset{2}{g}_{ab} = 2\gamma U\delta_{ab}, \tag{5b}$$

$$\begin{aligned}\overset{3}{g}_{0a} = &-\frac{1}{2}(3+4\gamma+\alpha_1-\alpha_2+\zeta_1-2\xi)V_a\\ &-\frac{1}{2}(1+\alpha_2-\zeta_1+2\xi)W_a,\end{aligned} \tag{5c}$$

$$\begin{aligned}\overset{4}{g}_{00} = &-2\beta U^2+(2+2\gamma+\alpha_3+\zeta_1-2\xi)\Phi_1\\ &+2(1+3\gamma-2\beta+\zeta_2+\xi)\Phi_2\\ &+2(1+\zeta_3)\Phi_3+2(3\gamma+3\zeta_4-2\xi)\Phi_4\\ &-2\xi\Phi_W-(\zeta_1-2\xi)\mathcal{A}.\end{aligned} \tag{5d}$$

For the last term $\overset{4}{g}_{ab}$ no such expansion is assumed in the standard PPN formalism, but it may be included in an extended formalism [13,14]. The terms on the right hand side are the so-called PPN potentials and PPN parameters; see Sects. 4.6 and 4.7 for their definition and [5] for a physical explanation. Here the PPN potentials describe the matter distribution, and their form is independent of the theory under consideration, while the PPN parameters are independent of the matter distribution, and their values are determined by the gravity theory. In order to determine the values of the PPN parameters for a given gravity theory, one follows a well-defined procedure to expand the gravitational field equations into velocity orders, which are then solved successively, up to the fourth order. The virtue of the PPN formalism is the fact that this form of the metric is sufficiently generic to solve the metric field equations of a large number of gravity theories, in which the source of gravity is the matter energy-momentum. In order to obtain this solution, one must also decompose the energy-momentum tensor (1) into space and time components, as well as into velocity orders. For the matter variables, one assigns the orders $\rho \sim \Pi \sim \mathcal{O}(2)$ and $p \sim \mathcal{O}(4)$, based on their order of magnitude in the solar system. Lowering the indices of the energy-momentum tensor, its components then take the form

$$\Theta_{00} = \rho\left(1+\Pi+|v|^2-\overset{2}{g}_{00}\right)+\mathcal{O}(6), \tag{6a}$$

$$\Theta_{0a} = -\rho v_a+\mathcal{O}(5), \tag{6b}$$

$$\Theta_{ab} = \rho v_a v_b + p\delta_{ab}+\mathcal{O}(6). \tag{6c}$$

Further, one assumes that the gravitational field is quasi-static, which means that it changes only due to the motion of the source matter, excluding, e.g., transient gravitational waves. Hence, time derivatives of all physical quantities are weighted with an additional velocity order, $\partial_0 \sim \mathcal{O}(1)$. In particular, this affects derivatives of the metric, which enter the field equations through the Levi–Civita connection

$$\overset{\circ}{\Gamma}{}^{\gamma}{}_{\alpha\beta} = \frac{1}{2}g^{\gamma\delta}(\partial_\alpha g_{\delta\beta}+\partial_\beta g_{\alpha\delta}-\partial_\delta g_{\alpha\beta}) \tag{7}$$

and its curvature tensor. Here and in the following, we denote terms related to the Levi–Civita connection with an empty circle, in order to distinguish them from other connections to be introduced later, following the standard convention in the literature on teleparallel and related gravity theories, where this distinction becomes relevant; note that this circle is a distinct symbol from a zero denoting the zeroth velocity order.

2.2 Tetrad extension

There exist numerous gravity theories in which one of the fundamental fields is a tetrad (or coframe) field $\theta^{\Gamma}{}_{\alpha}$, which defines the metric via the relation

$$g_{\alpha\beta} = \eta_{\Gamma\Delta}\theta^{\Gamma}{}_{\alpha}\theta^{\Delta}{}_{\beta}, \tag{8}$$

where $\eta_{\Gamma\Delta} = \mathrm{diag}(-1,1,1,1)$ is again the Minkowski metric, here with Lorentz indices. Its post-Newtonian expansion can be obtained as follows [15–17]. Its zeroth order is given by a diagonal background tetrad,

$$\overset{0}{\theta}{}^{\Gamma}{}_{\alpha} = \Delta^{\Gamma}{}_{\alpha} = \mathrm{diag}(1,1,1,1). \tag{9}$$

defined in the previous section. For any higher order terms $\overset{k}{\theta}{}^{\Gamma}{}_{\alpha}$ in its perturbative expansion, it turns out to be more convenient to work with the expressions [18,19]

$$\overset{k}{\tau}_{\alpha\beta} = \eta_{\alpha\gamma}\Delta_{\Gamma}{}^{\gamma}\overset{k}{\theta}{}^{\Gamma}{}_{\beta}, \tag{10}$$

which carry only spacetime indices, where

$$\Delta_{\Gamma}{}^{\alpha} = \mathrm{diag}(1,1,1,1) \tag{11}$$

is the inverse background tetrad. While it is entirely possible to use the perturbations $\overset{k}{\tau}_{\alpha\beta}$ as fundamental variables, it turns out to be more convenient to decompose them into metric perturbations and another, independent degree of freedom. This can be achieved by noting that the metric perturbations follow from the relation (8) to be given by

$$\overset{n}{g}_{\alpha\beta} = \eta_{\Gamma\Delta}\sum_{k=0}^{n}\overset{k}{\theta}{}^{\Gamma}{}_{\alpha}\overset{n-k}{\theta}{}^{\Delta}{}_{\beta} = \eta^{\gamma\delta}\sum_{k=0}^{n}\overset{k}{\tau}_{\gamma\alpha}\overset{n-k}{\tau}_{\delta\beta}. \tag{12}$$

Hence, the nth order metric perturbation encodes $\overset{n}{g}_{\alpha\beta}$ the symmetric part $2\overset{n}{\tau}_{(\alpha\beta)}$ of the tetrad perturbation, up to lower order terms. Isolating the highest orders from the sum on the right hand side, we find that this symmetric part is given by

$$\overset{n}{\tau}_{\alpha\beta}+\overset{n}{\tau}_{\beta\alpha} = \overset{n}{g}_{\alpha\beta}-\eta^{\gamma\delta}\sum_{k=1}^{n-1}\overset{k}{\tau}_{\gamma\alpha}\overset{n-k}{\tau}_{\delta\beta}. \tag{13}$$

For the antisymmetric part, which constitute the aforementioned independent degree of freedom, one may define another, antisymmetric tensor field by

$$\overset{n}{a}_{\alpha\beta} = 2\overset{n}{\tau}_{[\alpha\beta]} = \overset{n}{\tau}_{\alpha\beta} - \overset{n}{\tau}_{\beta\alpha}. \tag{14}$$

In summary, the perturbative orders $\overset{n}{\theta}{}^{\Gamma}{}_{\alpha}$ of the tetrad for $n \geq 1$ are then defined as

$$\overset{n}{\theta}{}^{\Gamma}{}_{\alpha} = \frac{1}{2}\eta^{\beta\gamma}\Delta^{\Gamma}{}_{\beta}\left(\overset{n}{g}_{\gamma\alpha} + \overset{n}{a}_{\gamma\alpha} - \eta_{\Delta\Theta}\sum_{k=1}^{n-1}\overset{k}{\theta}{}^{\Delta}{}_{\gamma}\overset{n-k}{\theta}{}^{\Theta}{}_{\alpha}\right) \tag{15}$$

in terms of $\overset{n}{g}_{\alpha\beta}$ and $\overset{n}{a}_{\alpha\beta}$. Following a similar line of argument as for the metric, the only non-vanishing components of the antisymmetric tensor up to the fourth velocity order which must be considered are given by

$$\overset{2}{a}_{ab}, \quad \overset{3}{a}_{0a}, \quad \overset{4}{a}_{ab}. \tag{16}$$

Note finally that the tetrad $\theta^{\Gamma}{}_{\alpha}$ comes also with an inverse, the frame field $e_{\Gamma}{}^{\alpha}$. It follows from its relation with the coframe field that its background is the diagonal element $\overset{0}{e}_{\Gamma}{}^{\alpha} = \Delta_{\Gamma}{}^{\alpha}$, while higher perturbation orders are recursively defined as

$$\overset{n}{e}_{\Gamma}{}^{\alpha} = -\Delta_{\Gamma}{}^{\beta}\sum_{k=0}^{n-1}\overset{k}{e}_{\Delta}{}^{\alpha}\overset{n-k}{\theta}{}^{\Delta}{}_{\beta}, \tag{17}$$

where the tetrad perturbations on the right hand side are further expanded using the rule (15).

2.3 Teleparallel geometry

The tetrad and its perturbative expansion discussed above play a fundamental role as the dynamical field variable in a particular class of gravity theories known as teleparallel gravity [20]. In their covariant formulation [21], another fundamental field variable is used, which is the spin connection $\overset{\bullet}{\omega}{}^{\Gamma}{}_{\Delta\alpha}$. Together with the tetrad and its inverse, it defines another connection, whose coefficients are given by[2]

$$\overset{\bullet}{\Gamma}{}^{\gamma}{}_{\alpha\beta} = e_{\Gamma}{}^{\gamma}\left(\partial_{\alpha}\theta^{\Gamma}{}_{\beta} + \overset{\bullet}{\omega}{}^{\Gamma}{}_{\Delta\alpha}\theta^{\Delta}{}_{\beta}\right). \tag{18}$$

The spin connection is further restricted to be flat and antisymmetric, so that also the affine connection has vanishing curvature and is metric-compatible,

$$\begin{aligned}&\partial_{\alpha}\overset{\bullet}{\Gamma}{}^{\gamma}{}_{\beta\delta} - \partial_{\beta}\overset{\bullet}{\Gamma}{}^{\gamma}{}_{\alpha\delta} + \overset{\bullet}{\Gamma}{}^{\gamma}{}_{\alpha\lambda}\overset{\bullet}{\Gamma}{}^{\lambda}{}_{\beta\delta}\\ &\quad -\overset{\bullet}{\Gamma}{}^{\gamma}{}_{\beta\lambda}\overset{\bullet}{\Gamma}{}^{\lambda}{}_{\alpha\delta} = 0, \quad \overset{\bullet}{\nabla}_{\gamma}g_{\alpha\beta} = 0,\end{aligned} \tag{19}$$

but it possesses non-vanishing torsion. Under these conditions, the spin connection turns out to be a gauge quantity, so that one can locally choose a Lorentz gauge in which it vanishes, known as the Weitzenböck gauge [22]. Choosing this gauge, $\overset{\bullet}{\omega}{}^{\Gamma}{}_{\Delta\alpha} \equiv 0$, the connecting coefficients (18) are expressed in terms of the tetrad and its inverse alone [18]. This allows to derive their perturbative expansion, and the perturbative expansion of any derived tensor fields, in terms of the perturbations (15) and (17), and hence in terms of $\overset{n}{g}_{\alpha\beta}$ and $\overset{n}{a}_{\alpha\beta}$.

2.4 Symmetric teleparallel geometry

Finally, the third class of gravity theories discussed here has become known as symmetric teleparallel gravity theories [23]. In these theories yet another connection is employed, whose coefficients we denote by $\overset{\times}{\Gamma}{}^{\gamma}{}_{\alpha\beta}$, and which is assumed to have vanishing curvature and torsion,

$$\begin{aligned}&\partial_{\alpha}\overset{\times}{\Gamma}{}^{\gamma}{}_{\beta\delta} - \partial_{\beta}\overset{\times}{\Gamma}{}^{\gamma}{}_{\alpha\delta} + \overset{\times}{\Gamma}{}^{\gamma}{}_{\alpha\lambda}\overset{\times}{\Gamma}{}^{\lambda}{}_{\beta\delta} - \overset{\times}{\Gamma}{}^{\gamma}{}_{\beta\lambda}\overset{\times}{\Gamma}{}^{\lambda}{}_{\alpha\delta} = 0,\\ &\overset{\times}{\Gamma}{}^{\gamma}{}_{\alpha\beta} - \overset{\times}{\Gamma}{}^{\gamma}{}_{\beta\alpha} = 0.\end{aligned} \tag{20}$$

It follows from these properties that there exists a local coordinate system (x'^{α}), called the *coincident gauge*, such that its connection coefficients $\overset{\times}{\Gamma}{}'^{\gamma}{}_{\alpha\beta}$ in this coordinate system vanish [24]. Hence, in the standard PPN coordinate system (x^{α}), the connection coefficients are given by

$$\overset{\times}{\Gamma}{}^{\gamma}{}_{\alpha\beta} = \frac{\partial x^{\gamma}}{\partial x'^{\delta}}\frac{\partial x'^{\delta}}{\partial x^{\alpha}\partial x^{\beta}}. \tag{21}$$

The post-Newtonian expansion of these connection coefficients is derived from the assumption that the two coordinate systems (x^{α}) and (x'^{α}) are related by an infinitesimal coordinate transformation, induced by a vector field ξ^{α}, so that up to the quadratic order one can write

$$x'^{\alpha} = x^{\alpha} + \xi^{\alpha} + \frac{1}{2}\xi^{\beta}\partial_{\beta}\xi^{\alpha} + \cdots \tag{22}$$

From this assumption follows that the connection coefficients take the form

$$\begin{aligned}\overset{\times}{\Gamma}{}^{\gamma}{}_{\alpha\beta} = {}&\partial_{\alpha}\partial_{\beta}\xi^{\gamma} + \frac{1}{2}\left(\xi^{\delta}\partial_{\alpha}\partial_{\beta}\partial_{\delta}\xi^{\gamma}\right.\\ &\left.+2\partial_{(\alpha}\xi^{\delta}\partial_{\beta)}\partial_{\delta}\xi^{\gamma} - \partial_{\alpha}\partial_{\beta}\xi^{\delta}\partial_{\delta}\xi^{\gamma}\right) + \cdots,\end{aligned} \tag{23}$$

also up to the quadratic order in the vector field ξ^{α}. Finally, the vector field is expanded in velocity orders. It turns out that in the PPN formalism the only non-vanishing components are given by [25]

$$\overset{2}{\xi}{}^{a}, \quad \overset{3}{\xi}{}^{0}, \quad \overset{4}{\xi}{}^{a}. \tag{24}$$

From these components the perturbative expansion of the connection coefficients $\overset{\times}{\Gamma}{}^{\gamma}{}_{\alpha\beta}$ is obtained.

[2] Here we follow the convention used by *xTensor*, in which the first lower index α on the connection coefficients $\overset{\bullet}{\Gamma}{}^{\gamma}{}_{\alpha\beta}$ is the "derivative" index.

3 Mathematical foundation

Two fundamental mathematical concepts form the basis of the PPN formalism detailed in the previous section, and so their implementation is an important part of any computational tool to address this formalism. Here we briefly discuss these concepts and their formulation we choose to implement in *xPPN*. This will serve as an explanation of the mathematical background for the following sections. We discuss the split of tensors on spacetime into their space and time parts in Sect. 3.1 and the perturbative expansion into velocity order in Sect. 3.2.

3.1 3 + 1 split of tensors

As explained in the previous section, one of the crucial steps in applying the PPN formalism is the $3+1$ decomposition of all tensorial quantities. A proper, geometric interpretation of this split can be given by regarding the spacetime manifold M_4, equipped with coordinates $(x^\alpha) = (t, x^a)$, as a product manifold $M_4 \cong T_1 \times S_3$, where t is the time coordinate on the manifold $T_1 \cong \mathbb{R}$, while (x^a) are the spatial coordinates on S_3. It follows that for every $t \in T_1$, one can find a map $i_t : S_3 \to M_4, (x^a) \mapsto (t, x^a)$, whose image in M_4 is called the spatial slice, or constant time hypersurface at time t. The collection of all spatial slices then forms a foliation of M_4.

While the construction appears abstract, it gives a clear interpretation of the $3+1$ split of tensor fields as follows. A vector field with components X^α on M_4 splits in the form

$$X^\alpha \partial_\alpha = X^0 \partial_0 + X^a \partial_a \tag{25}$$

into a temporal part $X^0 \partial_0$, which is parallel to the coordinate lines generated by the time coordinate t, and a spatial part $X^a \partial_a$, which is tangent to the spatial hypersurfaces. Under purely spatial coordinate transformations, X^0 behaves as a scalar, while X^a are the components of a vector field. For any fixed value of t, we may thus take the pullback along i_t to obtain a scalar function $i_t^* X^0$, as well as a vector field $i_t^*(X^a \partial_a)$, both now being tensor fields on S_3, hence functions of the spatial coordinates (x^a). The dependence of the components X^α on the time coordinate t is still present as a dependence on the choice of the spatial hypersurface, and t is regarded as a parameter which corresponds to this choice, instead of a coordinate.

The interpretation of time t as a parameter instead of a coordinate is, of course, purely mathematical, but it allows for a number of simplification when it comes to the implementation of the $3+1$ split in computer algebra. Most importantly for the PPN formalism, it allows to treat time derivatives ∂_0 essentially differently from spatial derivatives ∂_a. This is necessary, because in the PPN formalism time derivatives are weighted with an additional velocity order, which is not the case for spatial derivatives. Another advantage becomes apparent when considering tensors with symmetries. For example, considering an antisymmetric tensor field $A_{\alpha\beta}$, the $3+1$ split can be written as

$$\begin{aligned} A_{\alpha\beta} \mathrm{d}x^\alpha \otimes \mathrm{d}x^\beta = A_{00} \mathrm{d}t \otimes \mathrm{d}t + A_{0a} \mathrm{d}t \otimes \mathrm{d}x^a \\ + A_{a0} \mathrm{d}x^a \otimes \mathrm{d}t + A_{ab} \mathrm{d}x^a \otimes \mathrm{d}x^b, \end{aligned} \tag{26}$$

where $A_{00} = 0$ and $A_{0a} = -A_{a0}$. Hence, the independent components can be described by a single covector field A_{0a} and antisymmetric tensor field A_{ab} on S_3. Thus, the use of tensor symmetries in conjunction with the $3+1$ split allows to immediately discard vanishing components, which helps to simplify the calculation.

The outlined interpretation of the $3+1$ split and the decomposition of tensor fields on spacetime M_4 into tensor fields on space S_3, with an additional parameter dependence, and corresponding counting of velocity orders, are a central ingredient of *xPPN*. In Sect. 5.1 we explain how to select certain components from the $3+1$ decomposition of a tensor field, while Sect. 5.3 shows how to decompose arbitrary tensorial expressions. The implementation of these operations is outlined in Appendix A.1, which shows the internal representation of decomposed tensor fields, while in Sect. A.2 we schematically explain the decomposition algorithm and its treatment of tensor symmetries.

3.2 Perturbative expansion in velocity orders

Another important ingredient of the PPN formalism is the perturbative expansion of tensor fields into velocity orders. Similarly to the expansion (2) for the metric, also any other tensor field A is expanded as

$$A = \sum_{n=0}^{\infty} \overset{n}{A}, \tag{27}$$

where each term is of the corresponding velocity order $\overset{n}{A} \sim \mathcal{O}(n)$, and where we omit tensor indices for brevity. Note that this is different from the convention used, e.g., in the *xPert* package [12], where the perturbative expansion takes the form

$$A = \sum_{n=0}^{\infty} \frac{\varepsilon^n}{n!} \Delta^n[A], \tag{28}$$

and thus has the explicit form of a Taylor expansion in the perturbation parameter ε. We therefore list a few formulas which follow from the perturbative expansion (27). Most notably, for a tensor $A = A_1 \cdots A_N$ given as a product of N tensors $A_1, \ldots, A_N$, one has the nth order term given by

$$\overset{n}{A} = \sum_{k_1+\ldots+k_N=n} \prod_{i=1}^{N} \overset{k_i}{A}_i, \tag{29}$$

where the orders k_i of the factors A_i run over non-negative integers. Another important relation is the expansion of expressions $F = f(A)$, where f is a scalar, or more general a tensorial function. In this case a Taylor expansion of the function f around the background $\overset{0}{A}$ is used,

$$f(A) = \sum_{k=0}^{\infty} \frac{\left(A - \overset{0}{A}\right)^k}{k!} f^{(k)}\left(\overset{0}{A}\right), \tag{30}$$

where $f^{(k)}$ denotes the kth derivative, and then the product formula (29) is applied. For a function of N arguments, this formula naturally generalizes to

$$f(A_1, \ldots, A_N) = \sum_{k_1=0}^{\infty} \cdots$$

$$\sum_{k_N=0}^{\infty} f^{(k_1,\ldots,k_N)}\left(\overset{0}{A}_1, \ldots, \overset{0}{A}_N\right) \prod_{i=1}^{N} \frac{\left(A_i - \overset{0}{A}_i\right)^{k_i}}{k_i!}. \tag{31}$$

Finally, time derivatives are weighted with an additional velocity order, and so for $A' = \partial_0 A$ one has

$$\overset{n}{A}{}' = \partial_0 \overset{n-1}{A}, \tag{32}$$

and analogously for higher derivative orders, where the left hand side is to be understood as the nth order term in the expansion of A'.

In *xPPN*, the perturbative expansion of tensor fields into velocity orders, which takes into account the rules outlined above, is implemented in the function `VelocityOrder`, which is explained in detail in Sect. 5.4. A few notes on the implementation of this function are given in Appendix A.3.

4 Geometric objects defined by *xPPN*

The PPN formalism relies on the existence of a number of objects, such as the spacetime manifold, the physical metric and its background value, as well as a particular set of potentials and constant parameters, which are necessary for its application to any gravity theory. For convenience, these objects are already defined by *xPPN* when the package is loaded, so that the user must define only those additional geometric objects which are specific to the gravity theory under consideration. Here we give an overview of these predefined objects. The manifolds corresponding to the split of spacetime into space and time, as well as a number of bundles over these manifolds, are given in Sect. 4.1. To define tensors on these manifolds and write tensorial expressions, *xAct* relies on indices; Sect. 4.2 lists the indices defined by *xPPN* for the given bundles. The geometric objects constituting the background geometry are listed in Sect. 4.3, while Sect. 4.4 gives a detailed account of the dynamical geometry and its perturbative expansion. The energy-momentum tensor and its expansion in fluid variables is shown in Sect. 4.5. Finally, the objects constituting the standard PPN metric are the PPN potentials detailed in Sect. 4.6 and the PPN parameters listed in Sect. 4.7.

4.1 Manifolds and bundles

Tensors in *xTensor* are defined with respect to a manifold. In order to implement the 3 + 1 decomposition discussed in Sect. 3.1, *xPPN* defines three manifolds:

1. `MfTime` is the one-dimensional time manifold T_1.
2. `MfSpace` is the three-dimensional space manifold S_3. All tensors for which the 3 + 1 split into time and space components has been carried out are defined on this manifold, with an additional dependence on the time parameter `TimePar`.
3. `MfSpacetime` is the four-dimensional spacetime manifold $M_4 \cong T_1 \times S_3$. It is defined as a product manifold, so that sums over tensor indices of M_4 may automatically be decomposed into sums over T_1 and S_3. All geometric objects which appear in the theory under consideration should be defined on this manifold.

Each of these manifolds is canonically equipped with a tangent bundle; *xTensor* defines these automatically, and they are named `TangentMfTime`, `TangentMfSpace` and `TangentMfSpacetime`, respectively, and denoted by prefixing the manifold symbol with $\mathbb{T}$. In addition, *xPPN* defines another vector bundle over each of these three manifolds, whose rank is the same as the dimension of the manifold, and which is used to define quantities carrying Lorentz indices. These bundles are named `LorentzMfTime`, `LorentzMfSpace` and `LorentzMfSpacetime`, respectively, and denoted by prefixing the manifold symbol with $\mathbb{L}$.

4.2 Indices

Indices play an important role in *xTensor*, since they are used to denote tensor expressions and establish the relation between tensor slots and vector bundles. Often a large number of indices is necessary for longer tensor expressions, and each of them must be defined as a symbol (possibly with an appended number). *xPPN* addresses this problem by predefining a large number of indices using symbols which are unlikely to collide with other notation. In particular, the following indices are defined:

1. `LI[0]` is used by *xPPN* to denote the time component of the 3 + 1 decomposed forms of tensors, and is printed as 0. From the viewpoint of *xTensor*, this is a label index,

which is not associated with any vector bundle and has no implied meaning. It is used because 3 + 1 decompositions are defined on S_3, and so only spatial indices can be associated with the tangent bundle. Also it may appear an arbitrary number of times and in any position in any tensor expression.

2. `\[ScriptT]` (printed as t) is the generic index on the tangent bundle $\mathbb{T}T_1$. *xTensor* will use this to automatically generate as many indices $t_1, t_2, \dots$ as needed as an intermediate step when performing a 3 + 1 decomposition of indices, before replacing them with 0.
3. `\[ScriptCapitalT]` (printed as $\mathcal{T}$) is the generic index on the Lorentz bundle $\mathbb{L}T_1$. It is used by *xTensor* in the same way as t on $\mathbb{T}T_1$.
4. Lowercase Latin letters $a, \dots, z$ (character codes 97–122) are used to denote indices on $\mathbb{T}S_3$. *xPPN* automatically defines the symbols `T3a`, ..., `T3z` to denote these indices, so that there is no need for the user to manually declare any indices, while at the same time avoiding possible name conflicts and leaving single letters free for other use. The prefix "`T3`" is omitted when the symbol is printed in output form, so that only the index letter itself appears in printed output.
5. Uppercase Latin letters $A, \dots, Z$ (character codes 65–90) are used to denote indices on $\mathbb{L}S_3$. As for the lowercase indices listed in the previous item, *xPPN* defines symbols `L3A`, ..., `L3Z` for these uppercase indices, and omits the prefix "`L3`" when printing the indices in output form.
6. Lowercase Greek letters $\alpha, \dots, \omega$ (character codes 945–969, 977, 981, 982, 1008, 1009, 1013) are used to denote indices on $\mathbb{T}M_4$. The corresponding symbols defined by *xPPN* are denoted `T4α`, ..., `T4ω`.
7. Uppercase Greek letters Α, ..., Ω (character codes 913–929, 931–937) are used to denote indices on $\mathbb{L}M_4$. The corresponding symbols defined by *xPPN* are denoted `L4A`, ..., `L4A<W>`.

To obtain a list of pre-defined indices for any of these bundles, the *xTensor* command `IndicesOfVBundle` may be used. For example, to list all indices defined on $\mathbb{T}M_4$, use:

```
In[]:= IndicesOfVBundle[TangentMfSpacetime]
```

The Mathematica command `Names`, together with a string pattern, may also be used to list these indices. This will give a similar result as the aforementioned command:

```
In[]:= Names["T4" ~~ _]
```

4.3 Background geometry

xPPN automatically defines a number of geometric objects corresponding to the background geometry, around which the perturbative PPN expansion is performed. On each of the three manifolds T_1, S_3, M_4 a metric, a tetrad and an inverse tetrad are defined. They are denoted as given in Table 1. Each of these geometric objects is defined to be constant with respect to the partial derivative `PD` on its respective manifold, so that `PD` applied to any of these objects vanishes. Further, contractions of a tetrad and its inverse are carried out automatically, and yield the corresponding `delta` tensor.

NB! Note that each of the background metrics is defined as the first metric on its respective manifold. Hence, it is also used by *xTensor* in order to raise and lower indices, such that for a vector defined as A^α one has $A_\alpha A^\alpha = A^\alpha A^\beta \eta_{\alpha\beta}$, which can easily be verified by using the *xTensor* command `SeparateMetric`:

```
In[]:= DefTensor[A[T4α], {MfSpacetime}]

In[]:= ToCanonical[SeparateMetric[][A[−T4α] A[T4α]]]
```
Out[]= $A^\alpha A^\beta \eta_{\alpha\beta}$

In order to raise and lower indices with the physical metric $g_{\alpha\beta}$, the metric (as defined in the following section) *must* be written out explicitly.

4.4 Dynamical geometry

The background geometry detailed above is defined independently of any gravity theory and matter configuration. This is contrasted with the dynamical geometry which we discuss next. These are the dynamical fields which are used to mediate the gravitational interaction, and which are related to the matter source by the gravitational field equations. The most important of these geometric objects is the metric, which we discuss in Sect. 4.4.1. A similar role may be taken by the tetrad, as shown in Sect. 4.4.2. Further, three covariant derivatives are defined, which represent the three connections used in different formulations of general relativity [26]: the Levi–Civita connection in Sect. 4.4.3, the teleparallel connection in Sect. 4.4.4 and the symmetric teleparallel connection in Sect. 4.4.5.

4.4.1 Metric

As detailed in Sect. 2.1, the central object in the PPN formalism is the metric $g_{\alpha\beta}$, which is denoted by `Met[-T4α, -T4β]` in *xPPN*. Its inverse $g^{\alpha\beta}$ is written as `InvMet[T4α, T4β]`. Also here it must be emphasized that this is different from `Met[T4α, T4β]`, which would be interpreted differently:

Table 1 Background geometric objects defined by *xPPN*

Symbol	Definition	Manifold	Indices
`BkgMetricM4`	$\eta_{\alpha\beta} = \mathrm{diag}(-1, 1, 1, 1)$	M_4	$(-\mathbb{T}M_4, -\mathbb{T}M_4)$
`BkgMetricS3`	$\delta_{ab} = \eta_{ab}$	S_3	$(-\mathbb{T}S_3, -\mathbb{T}S_3)$
`BkgMetricT1`	$\eta_{00} = -1$	T_1	$(-\mathbb{T}T_1, -\mathbb{T}T_1)$
`BkgTetradM4`	$\Delta^\Gamma{}_\alpha = \mathrm{diag}(1, 1, 1, 1)$	M_4	$(\mathbb{L}M_4, -\mathbb{T}M_4)$
`BkgTetradS3`	$\Delta^A{}_a$	S_3	$(\mathbb{L}S_3, -\mathbb{T}S_3)$
`BkgTetradT1`	$\Delta^0{}_0$	T_1	$(\mathbb{L}T_1, -\mathbb{T}T_1)$
`BkgInvTetradM4`	$\Delta_\Gamma{}^\alpha = \mathrm{diag}(1, 1, 1, 1)$	M_4	$(-\mathbb{L}M_4, \mathbb{T}M_4)$
`BkgInvTetradS3`	$\Delta_A{}^a$	S_3	$(-\mathbb{L}S_3, \mathbb{T}S_3)$
`BkgInvTetradT1`	$\Delta_0{}^0$	T_1	$(-\mathbb{L}T_1, \mathbb{T}T_1)$

In[]:= ToCanonical[SeparateMetric[][Met[T4α, T4β]]]
Out[]= $\eta^{\alpha\gamma}\eta^{\beta\delta}g_{\gamma\delta}$

A number of rules are defined for the perturbative expansion of the metric, and automatically applied when this expansion is performed. The zeroth order reduces to the background Minkowski metric (3), and so its space and time components are given by

$$\overset{0}{g}_{00} = -1, \quad \overset{0}{g}_{0a} = 0, \quad \overset{0}{g}_{ab} = \delta_{ab}. \tag{33}$$

The remaining components, up to the fourth velocity order, are set to vanish, except for the components (4), for which no further rules are defined. These must be solved for in order to determine the PPN parameters. See Sect. 5.2 for more information how to apply these rules using the function `ApplyPPNRules`.

4.4.2 Tetrad

In order to implement the tetrad extension to the PPN formalism detailed in Sect. 2.2, *xPPN* defines the tetrad $\theta^\Gamma{}_\alpha$ as the object `Tet[L4Γ, -T4α]`, together with an appropriate set of rules to evaluate its perturbative expansion using the formula (15). The antisymmetric tensor field $a_{\alpha\beta}$ occurring in this expansion is denoted by `Asym[-T4α, -T4β]` in *xPPN*. Further, next to the tetrad, also its inverse $e_\Gamma{}^\alpha$ is defined in *xPPN*, which is entered as `InvTet[-L4Γ, T4α]`. A number of automatically applied rules are defined for these inverse tetrads, so that they are contracted with tetrads if possible, and derivatives are evaluated according to

$$e_\Gamma{}^\alpha\theta^\Gamma{}_\beta = \delta^\alpha_\beta, \quad e_\Gamma{}^\alpha\theta^\Delta{}_\alpha = \delta^\Delta_\Gamma,$$
$$\partial_\beta e_\Gamma{}^\alpha = -e_\Gamma{}^\gamma e_\Delta{}^\alpha\partial_\beta\theta^\Delta{}_\gamma. \tag{34}$$

Further, perturbations of the tetrad are evaluated following the formula (17), together with expanding the tetrad perturbation using the expansion (15).

4.4.3 Levi–Civita connection

Together with the metric `Met[-T4α, -T4β]`, *xPPN* defines its unique metric-compatible, torsion-free Levi–Civita connection, whose coefficients are defined from the metric by the well-known formula (7). The corresponding covariant derivative is entered as `CD[-T4α]`, and it is written in postfix notation using a semicolon ";", as well as using the symbol $\overset{\circ}{\nabla}$ in prefix notation, to distinguish it from other covariant derivatives introduced later. *xTensor* automatically defines a number of tensors for any covariant derivative. Most notably are the Christoffel "tensors" `ChristoffelCD[T4γ, -T4α, -T4β]`, which are defined as the difference between the connection coefficients $\overset{\circ}{\Gamma}{}^\gamma{}_{\alpha\beta}$ and the (vanishing) coefficients of a fiducial connection associated to the partial derivatives ∂_α with respect to the coordinates. Further, *xTensor* defines a number of curvature tensors, such as the Riemann, Ricci, Einstein and Weyl tensors. *xPPN* defines the corresponding perturbative expansions in terms of the metric perturbations for all tensor fields derived from the covariant derivative `CD[-T4α]`.

4.4.4 Teleparallel connection

In order to work with teleparallel gravity theories, as discussed in Sect. 2.3, *xPPN* defines also the teleparallel connection (18), along with its torsion tensor and their perturbative expansion. The corresponding covariant derivative is entered as `FD[-T4α]`, and is denoted $\overset{\bullet}{\nabla}$ in prefix notation and a bar "|" in postfix notation. Working in the Weitzenböck gauge implies that the connection coefficients `ChristoffelFD[T4γ, -T4α, -T4β]` take the simple form

$$\overset{\bullet}{\Gamma}{}^\gamma{}_{\alpha\beta} = e_\Gamma{}^\gamma\partial_\alpha\theta^\Gamma{}_\beta, \tag{35}$$

and their perturbative expansion is defined accordingly.

4.4.5 *Symmetric teleparallel connection*

Finally, to accommodate the PPN formalism for symmetric teleparallel theories of gravity shown in Sect. 2.4, *xPPN* provides yet another pre-defined connection, which is characterized by vanishing torsion and curvature, but which is not compatible with the metric $g_{\mu\nu}$. Its covariant derivative is entered as `ND[-T4α]` and denoted by the symbol $\overset{\times}{\nabla}$ in prefix notation, and a hash "#" in postfix notation. The perturbative expansion (23) of its connection coefficients is expressed in terms of a vector field ξ^{α}, which is defined under the name `Xi[T4α]` in *xPPN*. Its perturbative expansion is implemented such that the only non-vanishing components are the three components (24). Also for this connection, *xAct* automatically defines torsion and curvature tensors, but both are set to vanish identically. The only tensorial quantity describing this connection and its relation to the Levi–Civita connection is the nonmetricity, which is given by

$$Q_{\alpha\beta\gamma} = \overset{\times}{\nabla}_{\alpha} g_{\beta\gamma}. \tag{36}$$

In *xPPN*, the nonmetricity is called `NonMet[-T4α, -T4β, -T4γ]`, and its perturbative expansion is obtained from the definition above.

4.5 Energy–momentum variables

In the PPN formalism it is assumed that the source of gravity is the energy-momentum tensor $\Theta_{\alpha\beta}$ of a perfect fluid (1), which defined by *xPPN* is a tensor on M_4 denoted by `EnergyMomentum[-T4α, -T4β]`. Note that it is defined with lower indices. In order to raise the indices, the physical metric $g_{\alpha\beta}$ *must* be used explicitly; implicitly raising the indices by writing `EnergyMomentum[T4α, T4β]` would use the background metric $\eta_{\alpha\beta}$, and hence *not* give the correct result. Also note that we use the symbol Θ instead of the more conventional T in order to avoid confusion with the torsion tensor.

For convenience, *xPPN* also defines the trace-reversed energy-momentum tensor

$$\bar{\Theta}_{\alpha\beta} = \Theta_{\alpha\beta} - \frac{1}{2} g_{\alpha\beta} g^{\gamma\delta} \Theta_{\gamma\delta}, \tag{37}$$

denoted by `TREnergyMomentum[-T4α, -T4β]`, and which is likewise defined as a tensor on M_4.

In order to describe the 3+1 split of the energy-momentum tensor and its decomposition into velocity orders, *xPPN* further defines the following tensors on S_3 depending on `TimePar`, which represent the variables describing the perfect fluid:

1. `Density[]` is the rest mass density $\rho \sim \mathcal{O}(2)$.
2. `Pressure[]` is the pressure $p \sim \mathcal{O}(4)$.
3. `InternalEnergy[]` is the specific internal energy $\Pi \sim \mathcal{O}(2)$.
4. `Velocity[T3a]` is the velocity $v^a \sim \mathcal{O}(1)$.

When the energy–momentum tensor $\Theta_{\alpha\beta}$ is expanded in velocity orders as shown in Sect. 5.4, the relations

$$\begin{aligned} &\overset{2}{\Theta}_{00} = \rho, \quad \overset{4}{\Theta}_{00} = \rho\left(\Pi + |v|^2 - \overset{2}{g}_{00}\right), \\ &\overset{3}{\Theta}_{0a} = -\rho v_a, \quad \overset{4}{\Theta}_{ab} = \rho v_a v_b + p\delta_{ab}, \end{aligned} \tag{38}$$

which follow directly from the expansion (6), are applied, while all other components vanish.

4.6 Post-Newtonian potentials

The post-Newtonian potentials are a central ingredient to the PPN formalism. In *xPPN* they are defined as tensors on the space manifold S_3 with an additional time dependence. Most of them are scalars, but there are also vector and tensor potentials, which therefore carry indices in the tangent bundle $\mathbb{T}S_3$. In particular, the following PPN potentials are defined:

1. `PotentialChi[]` is the post-Newtonian superpotential

$$\chi(t, \vec{x}) = -\int d^3x' \rho(t, \vec{x}')|\vec{x} - \vec{x}'|. \tag{39}$$

2. `PotentialU[]` is the Newtonian potential

$$U(t, \vec{x}) = \int d^3x' \frac{\rho(t, \vec{x}')}{|\vec{x} - \vec{x}'|}. \tag{40}$$

3. `PotentialUU[-T3a, -T3b]` is the anisotropic potential

$$\begin{aligned} U_{ab}(t, \vec{x}) &= \int d^3x' \frac{\rho(t, \vec{x}')}{|\vec{x} - \vec{x}'|^3}(x_a - x'_a)(x_b - x'_b) \\ &= \chi_{,ab} - \frac{1}{2}\triangle\chi\delta_{ab}. \end{aligned} \tag{41}$$

4. `PotentialV[-T3a]` is the isotropic vector potential

$$V_a(t, \vec{x}) = \int d^3x' \frac{\rho(t, \vec{x}')v_a(t, \vec{x}')}{|\vec{x} - \vec{x}'|}. \tag{42}$$

5. `PotentialW[-T3a]` is the anisotropic vector potential

$$W_a(t, \vec{x}) = \int d^3x' \frac{\rho(t, \vec{x}')v_b(t, \vec{x}')(x_a - x'_a)(x_b - x'_b)}{|\vec{x} - \vec{x}'|^3}. \tag{43}$$

6. `PotentialPhi1[]` is the kinetic energy potential

$$\Phi_1(t,\vec{x}) = \int \mathrm{d}^3x' \frac{\rho(t,\vec{x}')v(t,\vec{x}')^2}{|\vec{x}-\vec{x}'|}. \tag{44}$$

7. `PotentialPhi2[]` is the gravitational self-energy potential

$$\Phi_2(t,\vec{x}) = \int \mathrm{d}^3x' \frac{\rho(t,\vec{x}')U(t,\vec{x}')}{|\vec{x}-\vec{x}'|}. \tag{45}$$

8. `PotentialPhi3[]` is the internal energy potential

$$\Phi_3(t,\vec{x}) = \int \mathrm{d}^3x' \frac{\rho(t,\vec{x}')\Pi(t,\vec{x}')}{|\vec{x}-\vec{x}'|}. \tag{46}$$

9. `PotentialPhi4[]` is the pressure potential

$$\Phi_4(t,\vec{x}) = \int \mathrm{d}^3x' \frac{p(t,\vec{x}')}{|\vec{x}-\vec{x}'|}. \tag{47}$$

10. `PotentialA[]` is the anisotropic kinetic potential

$$\mathcal{A}(t,\vec{x}) = \int \mathrm{d}^3x' \frac{\rho(t,\vec{x}')\left[v_a(t,\vec{x}')(x_a-x'_a)\right]^2}{|\vec{x}-\vec{x}'|^3}. \tag{48}$$

11. `PotentialB[]` is the potential related to change in velocity

$$\mathcal{B}(t,\vec{x}) = \int \mathrm{d}^3x' \frac{\rho(t,\vec{x}')}{|\vec{x}-\vec{x}'|}(x_a-x'_a)\frac{\mathrm{d}v_a(t,\vec{x}')}{\mathrm{d}t}. \tag{49}$$

12. `PotentialPhiW[]` is the Whitehead potential

$$\begin{aligned}\Phi_W(t,\vec{x}) = &\int \mathrm{d}^3x' \int \mathrm{d}^3x'' \rho(t,\vec{x}')\rho(t,\vec{x}'')\\ &\times \frac{x_a-x'_a}{|\vec{x}-\vec{x}'|^3}\left(\frac{x'_a-x''_a}{|\vec{x}-\vec{x}''|} - \frac{x_a-x''_a}{|\vec{x}'-\vec{x}''|}\right).\end{aligned} \tag{50}$$

Note that these integrals are not implemented directly in *xPPN*, which makes no explicit use of the coordinates apart from the assumption that partial derivatives of the tensors defining the background geometry vanish. However, they enter into the formalism indirectly, since they define the relations between the potentials and the source terms which are explained in detail in Sect. 5.6.

Table 2 Constants representing the PPN parameters in *xPPN*

Symbol	Parameter
`ParameterBeta`	β
`ParameterGamma`	γ
`ParameterAlpha1`	α_1
`ParameterAlpha2`	α_2
`ParameterAlpha3`	α_3
`ParameterZeta1`	ζ_1
`ParameterZeta2`	ζ_2
`ParameterZeta3`	ζ_3
`ParameterZeta4`	ζ_4
`ParameterXi`	ξ

4.7 Post-Newtonian parameters

The PPN parameters are represented in *xPPN* by objects which are declared as constants within *xTensor*, so that any derivatives acting on them vanish. A full list is given in Table 2. Note that assuming the PPN parameters to be constant, which is one of the standard assumptions of the PPN formalism, means that it cannot be directly applied to, e.g., theories with massive fields mediating the gravitational interaction, since in such theories the values of the PPN parameters depend on the distance between the source and the test mass; also time variation of PPN parameters depending on a cosmological background is not considered. The former may be included in a future version of *xPPN* by introducing additional Yukawa-type potentials, which depend on a mass determining the interaction scale [27,28]. The latter might be implemented by allowing for PPN parameters which depend on `TimePar` instead of being declared constant [29].

5 Utility functions

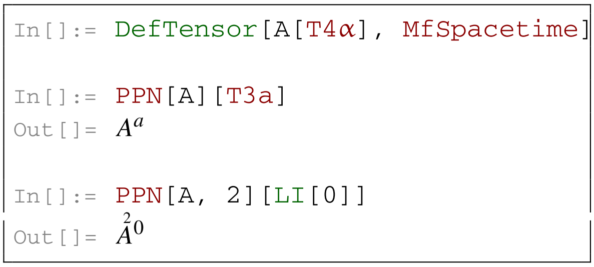
We now present a number of utility functions defined by *xPPN*, which can be used to manipulate terms which typically appear in the post-Newtonian expansion, perform necessary computational steps and solve the gravitational field equations in terms of the post-Newtonian potentials and parameters listed in the previous sections. Which of these functions need to be used highly depends on the gravity theory under consideration, and we show this for a few of them in an explicit example in Sect. 6; here we give an overview of the functions provided and how they are applied. In Sect. 5.1 we show how to refer to specific parts of tensors in their space-time decomposition and perturbative expansion. The definition and application of substitution rules for such terms is explained in Sect. 5.2. The decomposition of general tensorial expressions into space and time components is shown in Sect. 5.3, and further into velocity orders in Sect. 5.4. The transformation of terms using the Euler equations derived from energy-momentum conservation is displayed in Sect. 5.5, and applied to post-Newtonian potentials in Sect. 5.6. Finally, utility functions for sorting derivatives prior to applying these rules are shown in Sect. 5.7.

5.1 Selecting space-time components and velocity orders of tensors

As explained in Sect. 3.1, *xPPN* defines for each tensor declared on the spacetime manifold M_4 a number of tensors on the space manifold S_3, which represent the 3 + 1 decomposition of the initial tensor and its velocity orders. In order to access these components, *xPPN* defines the utility function `PPN` to easily obtain them from a tensor head, a sequence of indices and optionally a velocity order. This function can be used in two ways, taking the following arguments:

1. `PPN[`h`][`i`]`, where h is a tensor head and i is a sequence of indices, such that each index either belongs to S_3 or is `LI[0]` (possible with a minus sign);
2. `PPN[`h`,`n`][`i`]`, where n is a non-negative integer and h and i are as above.

The application of this function is illustrated by the following example of a vector A^α:

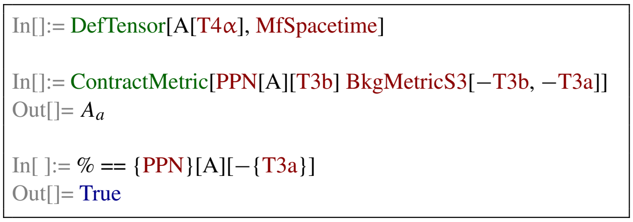


```
In[]:= DefTensor[A[T4α], MfSpacetime]

In[]:= PPN[A][T3a]
Out[]= A^a

In[]:= PPN[A, 2][LI[0]]
Out[]= \overset{2}{A}^0
```

Note that the indices do not have to be in the natural position in which they have been for the tensor head h. The function `PPN` yields the same result as if the indices had been specified in their natural position, and then raised or lowed with the background metric `BkgMetricS3` on S_3:

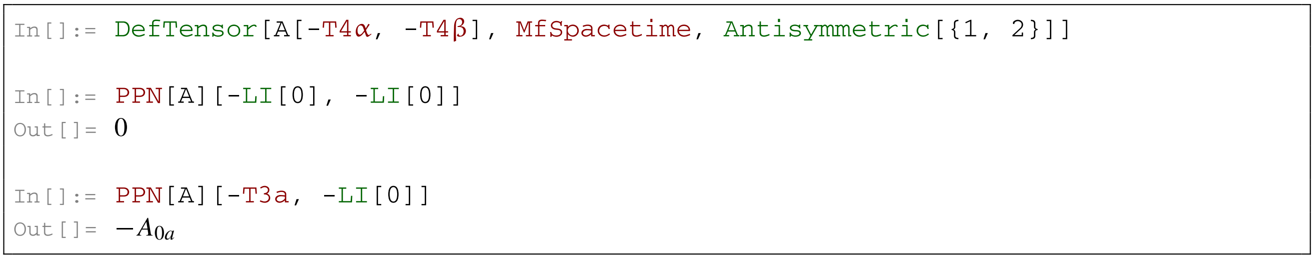


```
In[]:= DefTensor[A[T4α], MfSpacetime]

In[]:= ContractMetric[PPN[A][T3b] BkgMetricS3[−T3b, −T3a]]
Out[]= A_a

In[ ]:= % == {PPN}[A][−{T3a}]
Out[]= True
```

Tensor symmetries are taken into account, and indices are sorted into canonical order if possible:



```
In[]:= DefTensor[A[-T4α, -T4β], MfSpacetime, Antisymmetric[{1, 2}]]

In[]:= PPN[A][-LI[0], -LI[0]]
Out[]= 0

In[]:= PPN[A][-T3a, -LI[0]]
Out[]= −A_{0a}
```

5.2 Definition and application of replacement rules

For pre-defined tensors on the spacetime manifold M_4, such as the metric or the energy-momentum tensor, *xPPN* defines a number of substitution rules for the terms in their post-Newtonian expansion. In order to apply these rules to any given tensorial expression, *xPPN* provides the function `ApplyPPNRules`, which can be invoked in two different ways:

1. `ApplyPPNRules[`X`]` recursively applies the defined PPN rules to all tensors which appear in the expression X and its subexpressions.
2. `ApplyPPNRules[`X`,`h`]` applies the defined PPN rules only to those tensors within X with head h.

For example, the zeroth order of the physical metric is given by the Minkowski background metric:

```
In[]:= {PPN[Met, 0][−LI[0], −LI[0]], PPN[Met, 0][−LI[0], −T3a], PPN[Met, 0][−T3a, −T3b]}
```

Out[]= $\{\overset{0}{g}_{00}, \overset{0}{g}_{0a}, \overset{0}{g}_{ab}\}$

```
In[]:= ApplyPPNRules /@%
```

Out[]= $\{-1, 0, \delta_{ab}\}$

By default, *xPPN* does not associate any rules to the terms in the post-Newtonian expansion of user-defined tensors. Therefore, *xPPN* provides a number of functions to define and undefine such rules. This can be done with the following functions:

1. `OrderSet[PPN[h,n][i],X]` defines or replaces a rule, such that `ApplyPPNRules` substitutes the n'th order term `PPN[h,n][i]` by X, where X can be any tensorial expression which has the same free indices as defined by i.
2. `OrderUnset[PPN[h,n][i]]` removes the PPN rule associated with the term `PPN[h,n][i]` in the post-Newtonian expansion.
3. `OrderClear[h]` removes any PPN rules associated to the tensor head h.

Their application can be illustrated as follows.

```
In[]:= DefTensor[A[T4α], MfSpacetime]
In[]:= a = {PPN[A, 0][LI[0]], PPN[A, 0][T3a], PPN[A, 1][LI[0]], PPN[A, 1][T3a]};
In[]:= ApplyPPNRules /@ a
```

Out[]= $\{\overset{0}{A}^0, \overset{0}{A}^a, \overset{1}{A}^0, \overset{1}{A}^a\}$

```
In[]:= OrderSet[PPN[A, 0][LI[0]], 1];
In[]:= OrderSet[PPN[A, 0][T3a], 0];
In[]:= OrderSet[PPN[A, 1][T3a], Velocity[T3a]];
In[]:= ApplyPPNRules /@ a
```

Out[]= $\{1, 0, \overset{1}{A}^0, v^a\}$

```
In[]:= OrderUnset[PPN[A, 1][T3a]];
In[]:= ApplyPPNRules /@ a
```

Out[]= $\{1, 0, \overset{1}{A}^0, \overset{1}{A}^a\}$

```
In[]:= OrderClear[A];
In[]:= ApplyPPNRules /@ a
```

Out[]= $\{\overset{0}{A}^0, \overset{0}{A}^a, \overset{1}{A}^0, \overset{1}{A}^a\}$

5.3 3 + 1 space-time split of tensorial expressions

xPPN defines two functions handling the 3 + 1 decomposition of arbitrary tensorial expressions on spacetime into their space and time components, as explained in Sect. 3.1: `SpaceTimeSplit` and `SpaceTimeSplits`. While the former calculates one specific component of the 3+1 decomposition, the latter yields an array with all components. In both functions the decomposition is applied to both free and dummy indices.

The function `SpaceTimeSplit` takes two arguments: a tensor on M_4 (possibly containing free indices) and a list of replacement rules, which assigns to each free index in a bundle over M_4 either an index over the corresponding bundle over S_3 or the label index `LI[0]` (possibly dressed with a minus sign to indicate a lower index). For example, for an expression of the form $A^{\alpha}{}_{\beta}$ with free indices `T4α`, `-T4β`, the second argument may be any of the following:

1. `T4α →LI[0], -T4β→-LI[0]`
2. `T4α →T3a, -T4β→-LI[0]`
3. `T4α →LI[0], -T4β→-T3b`
4. `T4α →T3a, -T4β→-T3b`

Of course, other names than a, b may be used for the indices on S_3, but their position must remain the same; their role is to specify how the free indices in the resulting expression will be named. The function `SpaceTimeSplit` then calculates the component of the expression where each free index on M_4 is replaced by the specified indices of the 3+1 decomposition. For example, for a tensor $A^{\alpha}{}_{\beta}$ defined by

```
In[]:= DefTensor[A[T4α, −T4β], MfSpacetime]
```

one may use

```
In[]:= SpaceTimeSplit[A[T4α, -T4β], {T4α → T3a, -T4β → -LI[0]}]
Out[]= A^a_0

In[]:=% == {PPN}[A][{T3a}, -LI[0]]
Out[]= True
```

to obtain the component $A^{a}{}_{0}$. In contrast to the function `PPN`, which yields the $3+1$ decomposed component of a single tensor only, any tensorial expression may be decomposed by `SpaceTimeSplit` instead of a single tensor. For example, to decompose the expression $A^{\alpha}{}_{\gamma}A^{\gamma}{}_{\beta}$ one may use

```
In[]:= SpaceTimeSplit[A[T4α, −T4γ] A[T4γ, −T4β], {T4α → T3a, −T4β → −LI[0]}]
Out[]= A^a_0 A^0_0 + A^a_b A^b_0
```

Observe that also the dummy index γ has been split into a sum over dummy indices in the $3+1$ decomposition.

The function `SpaceTimeSplits` is similar, but in its second argument *only* indices of S_3 (and hence no `LI[0]`) are allowed as the right hand sides of the rules. The function then returns an array in which the free indices of the original expression are replaced either by the specified spatial indices or the time index 0, as shown in the following example:

```
In[]:= SpaceTimeSplits[A[T4α, -T4β], {T4α → T3a, -T4β → -T3b}]
Out[]= {{A^0_0, A^0_b}, {A^a_0, A^a_b}}
```

The order of the rules in the replacement list corresponds to the order of the dimensions of the resulting array. In the example above, the first dimension corresponds to α, while the second dimension corresponds to β. Hence,

```
In[]:= %[[1, 2]]
Out[]= A^0_b
```

selects the component where α is replaced by 0 (the first element of $(0, a)$), while β is replaced by b (the second element of $(0, b)$). Similarly,

```
In[]:= SpaceTimeSplits[A[T4α, −T4β], {−T4β → −T3b, T4α→ T3a}]
Out[]= {{A^0_0, A^a_0}, {A^0_b, A^a_b}}
```

yields the transposed array.

Finally, we remark that both `SpaceTimeSplit` and `SpaceTimeSplits` also operate on partial derivatives `PD[-T4α]` with respect to spacetime coordinates. These are converted as follows:

```
In[]:= DefTensor[A[], MfSpacetime]
In[]:= PD[−T4α][A[]]
Out[]= ∂_α A

In[]:= SpaceTimeSplits[%, {−T4α → −T3a}]
Out[]= {∂_0 A, ∂_a A}

In[]:= % == {ParamD[TimePar][PPN[A][]], PD[−T3a][PPN[A][]]}
Out[]= True
```

Observe that the time derivative is not represented as a partial derivative, but as a parameter derivative with respect to the time parameter `TimePar`. Finally, we remark that the automatic split is implemented only for partial derivatives `PD[-T4α]`; any other derivatives most be converted with `ChangeCovD` or `LieDToCovD`, as appropriate.

5.4 Decomposition into velocity orders

As explained in Sect. 3.2, an important part of the PPN formalism is the series expansion of expressions in velocity orders. In order to select a single term $\overset{n}{X} \sim \mathcal{O}(n)$ from an expression X, *xPPN* provides the function `VelocityOrder`. The term $\overset{n}{X}$ is then expressed as `VelocityOrder[`X`,`n`]`. Here n must be a non-negative integer, while X can be any tensorial expression on S_3, which may in addition depend on `TimePar`. It makes use of a number of standard relations for the velocity order in the PPN formalism: orders are distributed over products, and time derivatives are weighted with an additional velocity order. This is illustrated in the following example:

```
In[]:= DefTensor[A[T4α], MfSpacetime]
In[]:= VelocityOrder[PPN[A][T3a] PPN[A][-T3a], 2]
Out[]= \overset{0}{A}^a \overset{2}{A}_a + \overset{1}{A}^a \overset{1}{A}_a + \overset{2}{A}^a \overset{0}{A}_a

In[]:= VelocityOrder[ParamD[TimePar][PPN[A][-T3a]] + PD[-T3a][PPN[A][-LI[0]]], 3]
Out[]= \partial_0 \overset{2}{A}_a + \partial_a \overset{3}{A}_0
```

The function `VelocityOrder` supports the boolean option `UsePPNRules`. If it is set to `True` (the default case), any rules assigned to `PPNTensor` objects are applied immediately as soon as they are encountered. If it is set to `False`, no PPN rules are applied. For example, for the energy-momentum tensor this yields the following results:

```
In[]:= VelocityOrder[PPN[EnergyMomentum][-LI[0], -LI[0]], 2, UsePPNRules → True]
Out[]= ρ

In[]:= % == Density[]
Out[]= True

In[]:= VelocityOrder[PPN[EnergyMomentum][-LI[0], -LI[0]], 2, UsePPNRules → False]
Out[]= \overset{2}{\Theta}_{00}

In[]:= % == PPN[EnergyMomentum, 2][-LI[0], -LI[0]]
Out[]= True
```

Observe that in the first case the rule $\overset{2}{\Theta}_{00} \to \rho$ is applied, but not in the second case. For large expressions the default setting `UsePPNRules →True` is significantly faster, since the PPN rules imply the vanishing of many terms in the post-Newtonian expansion, which are thus immediately discarded when the rules are applied.

5.5 Euler equations

An important assumption of the PPN formalism is that the energy-momentum tensor $\Theta_{\alpha\beta}$ introduced in Sect. 4.5 satisfies the covariant conservation equation $\overset{\circ}{\nabla}_\alpha \Theta^{\alpha\beta} = 0$. Expanding this equation into velocity orders and performing a $3+1$ split into space and time components, one obtains the Euler equations, which govern the dynamics of the fluid. These equations are implemented in *xPPN* in three different functions, which perform the following substitutions on all matching subexpressions of their arguments:

1. `TimeRhoToEuler[`X`]` applies the replacement

$$\rho_{,0} \to -(\rho v_a)_{,a}. \tag{51}$$

2. `TimeVelToEuler[X]` applies the replacement

$$v_{a,0} \to \frac{1}{2}\overset{2}{g}_{00,a} - v_b v_{a,b} - \frac{p_{,a}}{\rho}. \tag{52}$$

3. `TimePiToEuler[X]` applies the replacement

$$\Pi_{,0} \to v_a\left(\frac{p_{,a}}{\rho} - \Pi_{,a} - \frac{1}{2}\overset{2}{g}_{00,a} - \frac{1}{2}\overset{2}{g}_{bb,a}\right) - \frac{p v_{a,a}}{\rho} - \frac{1}{2}\overset{2}{g}_{aa,0}. \tag{53}$$

The functions can be successively applied in order to transform terms involving higher order time derivatives.

5.6 Transformation of PPN potentials

The post-Newtonian potentials defined in Sect. 4.6 and their derivatives satisfy a number of relations among each other and with the energy-momentum variables defined in Sect. 4.5, some of which follow directly from the definition of the potentials, while others can be derived from the Euler equations discussed in the previous section. *xPPN* defines a number of utility functions in order to implement these relations and use them to turn PPN potentials and their derivatives into either other PPN potentials or terms constructed from the energy-momentum tensor. The most important of these is `PotentialToSource[X]`, which performs the following replacements on all subexpressions of X:

$$\begin{aligned}&\triangle\triangle\chi \to 8\pi\rho, \quad \triangle\triangle\mathcal{A} \to 8\pi(\rho v_a v_b)_{,ab} - 4\pi\triangle(\rho|v|^2),\\ &\triangle\triangle\mathcal{B} \to 8\pi[\triangle p - (U_{,a}\rho)_{,a}],\\ &\triangle\Phi_1 \to -4\pi\rho|v|^2, \quad \triangle\Phi_2 \to -4\pi\rho U,\\ &\triangle\Phi_3 \to -4\pi\rho\Pi, \triangle\Phi_4 \to -4\pi p,\\ &\triangle U \to -4\pi\rho, \quad \triangle V_a \to -4\pi\rho v_a,\\ &\triangle\Phi_W \to 4\pi\rho U - 4U_{,a}U_{,a} + 2U_{,ab}\chi_{,ab}.\end{aligned} \tag{54}$$

The remaining functions act on specific PPN potentials or their derivatives. Their effects are listed in Table 3.

5.7 Sorting of derivatives

In *xAct*, derivatives are, in general, not sorted automatically. This poses two difficulties, which may lead to problems in simplifications:

1. Mathematica does not recognize terms which are mathematically equal if they contain derivatives in different order. For example, expressions of the form $\partial_\alpha\partial_\beta X$ and $\partial_\beta\partial_\alpha X$ are equal, since partial derivatives commute, but since they are represented differently, Mathematica does not recognize this.

Table 3 Functions to transform specific PPN potentials

Function	Transformation
`PotentialChiToU`	$\triangle\chi \to -2U$
`PotentialUToChi`	$U \to -\frac{1}{2}\triangle\chi$
`PotentialUToUU`	$U \to U_{aa}$
`PotentialUUToU`	$U_{aa} \to U$
`PotentialUUToChi`	$U_{ab} \to \chi_{,ab} - \frac{1}{2}\triangle\chi\delta_{ab}$
`PotentialUToV`	$U_{,0} \to -V_{a,a}$
`PotentialUToW`	$U_{,0} \to W_{a,a}$
`PotentialVToU`	$V_{a,a} \to -U_{,0}$
`PotentialWToU`	$W_{a,a} \to U_{,0}$
`PotentialVToW`	$V_{a,a} \to -W_{a,a}$
`PotentialWToV`	$W_{a,a} \to -V_{a,a}$
`PotentialVToChiW`	$V_a \to W_a + \chi_{,0a}$
`PotentialWToChiV`	$W_a \to V_a - \chi_{,0a}$
`PotentialChiToPhiAB`	$\chi_{,00} \to \mathcal{A} + \mathcal{B} - \Phi_1$
`PotentialUToPhiAB`	$U_{,00} \to -\frac{1}{2}\triangle(\mathcal{A} + \mathcal{B} - \Phi_1)$

2. In order to apply the transformations listed in Sect. 5.6, pattern matching is applied to find divergences, time derivatives and Laplace operators acting on tensors. However, due to the way how derivatives are represented in *xTensor*, Mathematica recognizes such terms only if they are not interspersed with other derivatives. For example, $\partial_\alpha A^\alpha$ is found in $\partial_\beta\partial_\alpha A^\alpha$, but not in $\partial_\alpha\partial_\beta A^\alpha$, even though these terms are mathematically equal.

The first of these problems can be solved by defining a canonical order for derivatives and keeping them always sorted in canonical order. Such a canonical ordering is implemented as part of the *xTensor* function `ToCanonical`, so that the following terms are recognized by Mathematica as equal and canceled:

```
In[]:= DefTensor[A[], MfSpacetime]

In[]:= PD[−T4α][PD[−T4β][A[]]] − PD[−T4β][PD[−T4α][A[]]]
Out[]= ∂α∂βA − ∂β∂αA

In[]:= ToCanonical[%]
In[ ]:= % == PPN[A][−T3a]
Out[]= 0
```

However, this does not solve the second problem, since a different order of derivatives is required, depending on the type of pattern to be matched; for example, matching a time derivative would require the order $\partial_\alpha\partial_0 A^\alpha$, while matching a divergence would require the order $\partial_0\partial_\alpha A^\alpha$. Hence, keeping all indices in a fixed, canonical order would not allow for such patterns to be found. Hence, *xPPN* implements a different

```
In[]:= DefTensor[A[T4α], MfSpacetime]

In[]:= expr = PD[T3b][ParamD[TimePar][PD[−T3a][PD[−T3c][PD[−T3b][PPN[A][T3c]]]]]]
```
Out[]= $\partial^b \partial_0 \partial_a \partial_c \partial_b A^c$

```
In[]:= SortPDs[expr]
```
Out[]= $\partial_0 \partial_c \partial_b \partial^b \partial_a A^c$

```
In[]:= SortPDsToTime[expr, A]
```
Out[]= $\partial^b \partial_a \partial_c \partial_b \partial_0 A^c$

```
In[]:= SortPDsToDiv[expr, A]
```
Out[]= $\partial^b \partial_0 \partial_a \partial_b \partial_c A^c$

```
In[]:= SortPDsToBox[expr, A]
```
Out[]= $\partial_0 \partial_a \partial_c \partial_b \partial^b A^c$

method by defining different functions, which allow sorting derivatives on demand in any of the necessary orders. The following functions are defined:

1. `SortPDs[X]` sorts all derivatives in canonical order by applying the following rules:
   (a) Spatial derivatives are applied before time derivatives, i.e., moved to the right.
   (b) Spatial derivatives are sorted in lexicographic order, i.e., indices which come earlier in lexicographic order, are applied first.
   (c) Derivatives with upper indices are applied before derivatives with otherwise identical lower indices.
2. `SortPDsToTime[X,h]` sorts derivatives acting on tensors with head h such that time derivatives are applied first, i.e., moved to the right.
3. `SortPDsToDiv[X,h]` sorts derivatives acting on tensors with head h such that divergences may be matched, i.e., such that derivatives are applied first, whose index is contracted with a corresponding index of the tensor expression.
4. `SortPDsToBox[X,h]` sorts derivatives acting on tensors with head h such that Laplace operators are formed from spatial derivatives, if possible, and those are applied first.

The effect of these functions on an expression with multiple derivatives acting on a tensor is shown by the following code:

The implementation of these functions is inspired by the field theory package *xTras* [30], which defines a similar set of functions, which take into account also non-commuting derivatives by adding suitable correction terms. Also note that the replacement functions listed in Sect. 5.6 make use of these sorting functions automatically in order to establish the necessary order of indices before pattern matching is performed.

6 Example: scalar-tensor gravity

In order to demonstrate the use of the *xPPN* package, we provide a practical example in form of a complete, commented session, which calculates the PPN parameters of a scalar-tensor class of gravity theories, thereby showing the use of some of the functions detailed in the previous sections. The precise steps which are needed to determine the PPN parameters highly depend on the specific theory under consideration, and must be chosen accordingly. We briefly present the action and field equations of this class in Sect. 6.1. A few preliminary commands, for loading the package and setup, are given in Sect. 6.2. In section 6.3, the necessary tensors and constants for the calculation are defined. These are used in the definition of the field equations in Sect. 6.4. Their post-Newtonian expansion is derived in Sect. 6.5. In Sect. 6.6, the perturbative solution of the resulting equations is obtained. Finally, in Sect. 6.7, the PPN metric and parameters are calculated.

6.1 Action and field equations

In the following we discuss a class of scalar–tensor theories of gravity, whose action is given by [31]

$$S = \frac{1}{2\kappa^2}\int_{M_4} d^4x\sqrt{-g}\left(\psi R - \frac{\omega(\psi)}{\psi}\partial_\rho\psi\partial^\rho\psi\right) \\ +S_m[g_{\mu\nu}, \chi] \tag{55}$$

in Brans–Dicke like parametrization in the Jordan conformal frame. Here S_m denotes the matter part of the action, where we collectively denoted by χ the set of matter fields. The gravitational part contains a free function ω of the scalar field ψ. Each theory of this class is defined by a particular choice of this free function ω. By variation of this action with respect to the metric and the scalar field as well as subtraction of a suitable multiple of the trace of the metric field equation one obtains the field equations

$$\psi R_{\mu\nu} - \mathring{\nabla}_\mu\partial_\nu\psi - \frac{\omega}{\psi}\partial_\mu\psi\partial_\nu\psi + \frac{g_{\mu\nu}}{4\omega+6}\frac{d\omega}{d\psi}\partial_\rho\psi\partial^\rho\psi \\ = \kappa^2\left(\Theta_{\mu\nu} - \frac{\omega+1}{2\omega+3}g_{\mu\nu}\Theta\right), \tag{56a}$$

$$(2\omega+3)\mathring{\Box}\psi + \frac{d\omega}{d\psi}\partial_\rho\psi\partial^\rho\psi = \kappa^2\Theta, \tag{56b}$$

where $\mathring{\Box} = g^{\mu\nu}\mathring{\nabla}_\mu\mathring{\nabla}_\nu$ is the d'Alembert operator on the physical spacetime M_4.

6.2 Package loading and preliminaries

In order to load and use *xPPN*, as the first prerequisite a working installation of *xAct* [10] is needed. The files provided by *xAct* then reside in a directory named `xAct` in the Mathematica package search path. The *xPPN* package comes as a directory named `xPPN`, which must be placed inside the `xAct` directory. If both packages are installed correctly, *xPPN* can be loaded with the command

```
In[]:= << xAct‘xPPN‘
```

This should load *xPPN* and its dependencies from the *xAct* package suite. Note that loading may take some time, since *xPPN* calculates the perturbative expansion of the predefined tensor fields upon package loading. To suppress $ symbols in the index notation, it is useful to set the following *xAct* printing option:

```
In[]:= $PrePrint = ScreenDollarIndices;
```

Finally, we define two utility functions which help to create rules from equations; this functionality, which is simply a shorthand notation for the versatile function `MakeRule` from the *xTensor* package, are not specific to *xPPN*, and are defined here only to shorten the notation in the later course of this example:

```
In[]:= mkrg[eq_Equal] := MakeRule[Evaluate[List @@ eq],
    MetricOn → All, ContractMetrics → True]

In[]:= mkr0[eq_Equal] := MakeRule[Evaluate[List @@ eq],
    MetricOn → None, ContractMetrics → False]
```

6.3 Object definitions

We continue by defining the *xAct* objects which we will be using for the calculation and their notation. We start with the scalar field ψ, which is a tensor without any indices.

```
In[]:= DefTensor[psi[], MfSpacetime, PrintAs → "ψ"]
```

We then continue with the cosmological background value Ψ of the scalar field. This is a constant.

```
In[]:= DefConstantSymbol[psi0, PrintAs → "Ψ"]
```

Another constant which we will need is the gravitational constant κ.

```
In[]:= DefConstantSymbol[kappa, PrintAs → "κ"]
```

The action (55) also depends on a free function ω. This is defined as a scalar function.

```
In[]:= DefScalarFunction[omega, PrintAs → "ω"]
```

We then come to the field equations (56), which we write in the form $\mathcal{E}_{\alpha\beta} = 0$ and $\mathcal{E} = 0$, by moving the energy-momentum tensor to the left hand side. The two tensors representing these field equations are then defined as follows.

```
In[]:= DefTensor[MetEq[-T4α, -T4β], MfSpacetime, Symmetric[{1, 2}], PrintAs → "ℰ"]
In[]:= DefTensor[ScalEq[], MfSpacetime, PrintAs → "ℰ"]
```

Finally, in this example we will solve the field equations by making an ansatz for the solution in terms of PPN potentials and unknown, constant coefficients, which we will then determine. For this purpose, we define a function which creates such constant coefficients on demand as follows.

```
In[]:= aa[i_] := Module[{sym = Symbol["a" <> ToString[i]]},
    If[!ConstantSymbolQ[sym],
        DefConstantSymbol[sym, PrintAs → StringJoin["\!\(a\_", ToString[i], "\)"]]
    ];
    Return[sym]]
```

6.4 Field equations

In the next step, we enter the field equations (56). As mentioned before, we must pay attention to explicitly use the inverse metric $g^{\alpha\beta}$ for terms such as the kinetic term $\partial_\rho\psi\partial^\rho\psi$ of the scalar field or the trace $\Theta^\rho{}_\rho$ of the energy-momentum tensor. Taking these into account, we can enter the metric field equation (56a) as follows.

```
In[]:= psi[] ∗ RicciCD[−T4α, −T4β] − CD[−T4α][CD[−T4β][psi[]]] −
    PD[−T4α][psi[]] ∗ PD[−T4β][psi[]] ∗ omega[psi[]] / psi[] +
    InvMet[T4γ, T4δ] ∗ PD[−T4γ][psi[]] ∗ PD[−T4δ][psi[]] ∗ Met[−T4α, −T4β] ∗
    omega'[psi[]] / (4 omega[psi[]] + 6) −
    (EnergyMomentum[−T4α, −T4β] − InvMet[T4γ, T4δ] ∗ EnergyMomentum[−T4γ, −T4δ] ∗
    Met[−T4α, −T4β] ∗ (omega[psi[]] + 1) / (2 omega[psi[]] + 3)) ∗ kappa^2;

In[]:= meteqdef = MetEq[−T4α, −T4β] == %;
In[]:= meteqru = mkr0[meteqdef];
```

Similarly, we continue with the scalar field equation (56b):

```
In[]:= (2 omega[psi[]] + 3) * InvMet[T4α, T4β] * CD[-T4α][CD[-T4β][psi[]]] +
    omega'[psi[]] * InvMet[T4α, T4β] * PD[-T4α][psi[]] * PD[-T4β][psi[]] -
    kappa^2 * InvMet[T4α, T4β] * EnergyMomentum[-T4α, -T4β];

In[]:= scaleqdef = ScalEq[] == %;
In[]:= scaleqru = mkr0[scaleqdef];
```

6.5 Post-Newtonian expansion

The most crucial and resource intensive part of the calculation is the derivation of the post-Newtonian expansion of the field equations. For this purpose, we must first define the expansion of the scalar field. Here we impose the relations

$$\overset{0}{\psi} = \Psi, \quad \overset{1}{\psi} = 0, \quad \overset{3}{\psi} = 0. \tag{57}$$

In *xPPN*, they are implemented as follows:

```
In[]:= OrderSet[PPN[psi, 0][], psi0];
In[]:= OrderSet[PPN[psi, 1][], 0];
In[]:= OrderSet[PPN[psi, 3][], 0];
```

With these definitions in place, we can continue with the calculation. We will do so in two steps. First, we perform a 3 + 1 decomposition of the field equations. For this purpose, we convert the Levi–Civita covariant derivatives to partial derivatives and Christoffel symbols. Also we perform a number of simplifications. Again we start with the metric field equations.

```
In[]:= {#, # /. meteqru} &[MetEq[−T4α, −T4β]];
In[]:= ChangeCovD[%, CD, PD];
In[]:= Expand[%];
In[]:= SpaceTimeSplits[#, {−T4α → −T3a, −T4β → −T3b}] & /@ %;
In[]:= Expand[%];
In[]:= Map[ToCanonical, %, {3}];
In[]:= Map[SortPDs, %, {3}];
In[]:= meteq31list = %;
In[]:= meteq31def = Union[Flatten[MapThread[Equal, %, 2]]];
In[]:= meteq31ru = Flatten[mkrg /@ %];
```

We proceed similarly with the scalar field equations:

```
In[]:= {#, # /. scaleqru} &[ScalEq[]];
In[]:= ChangeCovD[%, CD, PD];
In[]:= Expand[%];
In[]:= SpaceTimeSplit[#, {}] & /@ %;
In[]:= Expand[%];
In[]:= ToCanonical /@ %;
In[]:= SortPDs /@ %;
In[]:= scaleq31list = %;
In[]:= scaleq31def = Equal @@ %;
In[]:= scaleq31ru = Flatten[mkrg[%]];
```

In the second step, we further decompose the equations obtained in the previous step into velocity orders $\mathcal{O}(0), \ldots, \mathcal{O}(4)$. Also here we perform a few tensor simplifications alongside the calculation, starting again with the metric equation.

```
In[]:= Outer[VelocityOrder, meteq31list, Range[0, 4]];
In[]:= Map[NoScalar, %, {4}];
In[]:= Expand[%];
In[]:= Map[ContractMetric[#, OverDerivatives → True,
    AllowUpperDerivatives → True] &, %, {4}];
In[]:= Map[ToCanonical, %, {4}];
In[]:= Map[SortPDs, %, {4}];
In[]:= meteqvlist = Simplify[%];
In[]:= meteqvdef = Union[Flatten[MapThread[Equal, %, 3]]]
In[]:= meteqvru = Flatten[mkrg /@ %];
```

Finally, we apply the same step also to the scalar field equation.

```
In[]:= Outer[VelocityOrder, scaleq31list, Range[0, 4]];
In[]:= Map[NoScalar, %, {2}];
In[]:= Expand[%];
In[]:= Map[ContractMetric[#, OverDerivatives → True,
    AllowUpperDerivatives → True] &, %, {2}];
In[]:= Map[ToCanonical, %, {2}];
In[]:= Map[SortPDs, %, {2}];
In[]:= scaleqvlist = Simplify[%];
In[]:= scaleqvdef = Flatten[MapThread[Equal, %, 1]]
In[]:= scaleqvru = Flatten[mkrg /@ %];
```

6.6 Perturbative solution

We can now use the post-Newtonian expansion of the field equations and solve them order by order. For this purpose we make use of prior knowledge that the field equations of the theory at hand can be solved by a particular ansatz for the metric and scalar field, so that at every step of the calculation we are left with solving for a number of constant coefficients; this ansatz and solution method must be adapted if the package is applied to other theories. We show the zeroth velocity order (the background vacuum solution) in Sect. 6.6.1, the second velocity order in Sect. 6.6.2, the third velocity order in Sect. 6.6.3 and the fourth velocity order in Sect. 6.6.4.

6.6.1 Zeroth velocity order

First, we verify that the background vacuum equations are solved identically by the background geometry. This can be checked as follows.

```
In[]:= PPN[MetEq, 0][−LI[0], −LI[0]] /. meteqvru
Out[]= 0

In[]:= PPN[MetEq, 0][−T3a, −T3b] /. meteqvru
Out[]= 0

In[]:= PPN[ScalEq, 0][] /. scaleqvru
Out[]= 0
```

6.6.2 Second velocity order

We then continue with the second velocity order. First, we gather the equations $\overset{2}{\mathcal{E}}_{00}$, $\overset{2}{\mathcal{E}}_{ab}$, $\overset{2}{\mathcal{E}}$ to be solved:

```
In[]:= eqs2 = FullSimplify[{PPN[MetEq, 2][-LI[0], -LI[0]],
    PPN[MetEq, 2][-T3a, -T3b], PPN[ScalEq, 2][]} /. meteqvru /. scaleqvru];
```

These equations take the form

$$\begin{aligned}
\overset{2}{\mathcal{E}} &= \kappa^2\rho + (2\omega(\Psi)+3)\triangle\overset{2}{\psi},\\
\overset{2}{\mathcal{E}}_{00} &= -\kappa^2\rho\frac{\omega(\Psi)+2}{2\omega(\Psi)+3} - \frac{\Psi}{2}\triangle\overset{2}{g}_{00},\\
\overset{2}{\mathcal{E}}_{ab} &= -\kappa^2\rho\frac{\omega(\Psi)+1}{2\omega(\Psi)+3}\delta_{ab} + \frac{\Psi}{2}\Big(\overset{2}{g}_{00,ab} - \overset{2}{g}_{cc,ab}\\
&\quad +2\overset{2}{g}_{c(a,b)c} - \triangle\overset{2}{g}_{ab}\Big) - \overset{2}{\psi}_{,ab}.
\end{aligned}\tag{58}$$

The easiest way to solve these equations is using an ansatz for the metric and scalar field perturbations in terms of the post-Newtonian potentials, with arbitrary, constant coefficients. This ansatz can be defined as follows:

Inspecting the obtained equations shows that they are given by

$$\begin{aligned}
\overset{2}{\mathcal{E}} &= \kappa^2\rho - 4\pi a_4(2\omega(\Psi)+3)\rho,\\
\overset{2}{\mathcal{E}}_{00} &= 2\pi\Psi a_1\rho - \kappa^2\rho\frac{\omega(\Psi)+2}{2\omega(\Psi)+3},\\
\overset{2}{\mathcal{E}}_{ab} &= \left(\frac{a_4}{2} + \frac{a_2+a_3-a_1}{4}\Psi\right)\triangle\chi_{,ab}\\
&\quad + \left[2\pi\Psi(a_2+a_3) - \kappa^2\frac{\omega(\Psi)+1}{2\omega(\Psi)+3}\right]\rho\delta_{ab}.
\end{aligned}\tag{59}$$

```
In[]:= ans2def = {PPN[Met, 2][−LI[0], −LI[0]] == aa[1] ∗ PotentialU[],
    PPN[Met, 2][−T3a, −T3b] == aa[2] ∗ PotentialU[] ∗ BkgMetricS3[−T3a, −T3b] +
        aa[3] ∗ PotentialUU[−T3a, −T3b],
    PPN[psi, 2][] == aa[4] ∗ PotentialU[]}
```

Out[]= $\left\{\overset{2}{g}_{00} = a_1 U\,,\quad \overset{2}{g}_{ab} = a_2 U\delta_{ab} + a_3 U_{ab}\,,\quad \overset{2}{\psi} = a_4 U\right\}$

```
In[]:= ans2ru = Flatten[mkrg /@ ans2def];
```

To obtain the equations to be solved, one inserts this ansatz into the field equations. Since the field equations contain derivatives of the metric and scalar field perturbations, this will yield derivatives of the post-Newtonian potentials U and U_{ab}. These must be matched with the terms arising from the energy-momentum tensor $\Theta_{\alpha\beta}$. For this purpose, one applies the functions listed in Sect. 5.6. For the potentials at hand it is most convenient to first convert them to derivatives of the superpotential χ, before transforming them to terms involving the matter density ρ.

These equations are solved if and only if the coefficients of each of the terms ρ, $\rho\delta_{ab}$ and $\triangle\chi_{,ab}$ vanishes individually. This yields four equations for the four coefficients $a_1, \ldots, a_4$. However, one finds that these are not linearly independent, due to the gauge freedom arising from the dif-

```
In[]:= eqs2 /. ans2ru;
In[]:= PotentialUToChi /@ %;
In[]:= PotentialUUToChi /@ %;
In[]:= Expand[%];
In[]:= ToCanonical /@ %;
In[]:= ContractMetric[#, OverDerivatives → True, AllowUpperDerivatives → True] & /@ %;
In[]:= PotentialToSource /@ %;
In[]:= Expand[%];
In[]:= ToCanonical /@ %;
In[]:= SortPDs /@ %;
In[]:= eqsa2 = FullSimplify[%];
```

feomorphism invariance of the theory. Hence, they must be supplemented with an additional, gauge fixing equation. The common choice in the PPN formalism is to eliminate the term U_{ab} from the component $\overset{2}{g}_{ab}$, thus setting $a_3 = 0$. We can thus determine the component equations:

```
In[]:= eqsc2 = FullSimplify[{
    Coefficient[eqsa2[[1]], Density[]], Coefficient[eqsa2[[3]], Density[]],
    Coefficient[eqsa2[[2]], Density[] * BkgMetricS3[-T3a, -T3b]], aa[3]}];
```

We can then solve the equations:

```
In[]:= sola2 = FullSimplify[First[Solve[# == 0 & /@ eqsc2, aa /@ Range[1, 4]]]];
```

This yields the solution

$$a_1 = \kappa^2 \frac{\omega(\Psi) + 2}{2\pi\Psi(2\omega(\Psi) + 3)},$$
$$a_2 = \kappa^2 \frac{\omega(\Psi) + 1}{2\pi\Psi(2\omega(\Psi) + 3)},$$
$$a_3 = 0, \quad a_4 = \frac{\kappa^2}{4\pi(2\omega(\Psi) + 3)}. \tag{60}$$

We may check that this indeed solves the equations:

```
In[]:= Simplify[eqsa2 /. sola2]
Out[]= {0, 0, 0}
```

Together with the ansatz we defined, this solution now yields the solution for the perturbations $\overset{2}{g}_{00}, \overset{2}{g}_{ab}, \overset{2}{\psi}$:

```
In[]:= sol2def = ans2def /. sola2;
In[]:= sol2ru = Flatten[mkrg /@ sol2def];
```

Finally, we also check that this result solves the second-order field equations:

```
In[]:= eqs2 /. sol2ru;
In[]:= Expand[%];
In[]:= PotentialToSource /@ %;
In[]:= ToCanonical /@ %;
In[]:= SortPDs /@ %;
In[]:= Simplify[%]
Out[]= {0, 0, 0}
```

6.6.3 Third velocity order

We then come to the third velocity order. Also here we proceed in full analogy to the second velocity order shown above. First, we isolate the field equation $\overset{3}{\mathcal{E}}_{0a}$ which we will solve:

```
In[]:= eqs3 = FullSimplify[PPN[MetEq, 3][-LI[0], -T3a] /. meteqvru];
```

This equation takes the form

$$\overset{3}{\mathcal{E}}_{0a} = \kappa^2 \rho v_a - \overset{2}{\psi}_{,0a} + \frac{\Psi}{2}\left(\overset{3}{g}_{0b,ab} - \triangle\overset{3}{g}_{0a} + \overset{2}{g}_{ab,0b} - \overset{2}{g}_{bb,0a}\right). \tag{61}$$

We then choose an ansatz for the third-order metric perturbation $\overset{3}{g}_{0a}$. This now involves the post-Newtonian vector potentials V_a and W_a with constant coefficients:

```
In[]:= ans3def = PPN[Met, 3][-LI[0], -T3a] ==
    aa[5] * PotentialV[-T3a] + aa[6] * PotentialW[-T3a]
Out[]= 3g0a = a5 Va + a6 Wa

In[]:= ans3ru = mkrg[ans3def];
```

To obtain the equation to be solved, we insert this ansatz into the field equation, together with the second-order solution obtained in the previous step. Also here we obtain derivatives acting on the post-Newtonian potentials, which we can simplify by using their interrelations, and finally convert them to terms matching the energy–momentum source. Here it is most useful to first eliminate the potential W_a. The resulting terms can then be converted to source terms and terms involving derivatives acting on U only:

```
In[]:= eqs3 /. ans3ru /. sol2ru;
In[]:= PotentialWToChiV[%];
In[]:= Expand[%];
In[]:= ContractMetric[%, OverDerivatives → True, AllowUpperDerivatives → True];
In[]:= PotentialChiToU[%];
In[]:= PotentialVToU[%];
In[]:= PotentialToSource[%];
In[]:= ToCanonical[%];
In[]:= SortPDs[%];
In[]:= eqsa3 = FullSimplify[%];
```

Inspecting this equation shows the following form:

$$\overset{3}{\mathcal{E}}_{0a} = [\kappa^2 + 2\pi\Psi(a_5 + a_6)]\left(\rho v_a - \frac{U_{,0a}}{4\pi}\right). \tag{62}$$

This equation only determines the sum $a_5 + a_6$ of the coefficients we introduced. Here we leave their difference as an undetermined parameter, which we will solve for when we come to the fourth velocity order, which will fix the post-Newtonian gauge. We thus determine the solution:

```
In[]:= sola3 = FullSimplify[
    First[Solve[{eqsa3 == 0, aa[6] - aa[5] == aa[0]}, {aa[5], aa[6]}]]];
```

The solution is given by

$$a_5 = -\frac{a_0}{2} - \frac{\kappa^2}{4\pi\Psi}, \quad a_6 = \frac{a_0}{2} - \frac{\kappa^2}{4\pi\Psi}. \tag{63}$$

We check that this solves the equations:

```
In[]:= Simplify[eqsa3 /. sola3]
Out[]= 0
```

Together with the ansatz for the third-order metric component $\overset{3}{g}_{0a}$ we obtain the solution:

```
In[]:= sol3def = ans3def /. sola3;
In[]:= sol3ru = mkrg[sol3def];
```

Finally, we also check that this solves the third-order field equation we started with:

```
In[]:= eqs3 /. sol2ru /. sol3ru;
In[]:= PotentialWToChiV[%];
In[]:= Expand[%];
In[]:= ContractMetric[%, OverDerivatives → True, AllowUpperDerivatives → True];
In[]:= PotentialChiToU[%];
In[]:= PotentialVToU[%];
In[]:= PotentialToSource[%];
In[]:= ToCanonical[%];
In[]:= SortPDs[%];
In[]:= Simplify[%]
Out[]= 0
```

6.6.4 Fourth velocity order

Finally, we come to the fourth velocity order, which is the most involved. For the theory under consideration, it is sufficient to solve the field equation $\overset{4}{\mathcal{E}}_{00}$, as it determines the necessary component $\overset{4}{g}_{00}$. We hence isolate this equation:

```
In[]:= eqs4 = PPN[MetEq, 4][-LI[0], -LI[0]] /. meteqvru;
```

We find that it takes the following form:

$$\begin{aligned}\overset{4}{\mathcal{E}}_{00} = &-\frac{\Psi}{4}\left(2\triangle\overset{4}{g}_{00} - 4\overset{3}{g}_{0a,0a} + 2\overset{2}{g}_{aa,00} + \overset{2}{g}_{00,a}\overset{2}{g}_{00,a} + \overset{2}{g}_{00,a}\overset{2}{g}_{bb,a}\right.\\ &\left.-2\overset{2}{g}_{00,a}\overset{2}{g}_{ab,b} - \overset{2}{g}_{00,ab}\overset{2}{g}_{ab}\right) - \frac{\omega'(\Psi)}{4\omega(\Psi)+6}\overset{2}{\psi}_{,a}\overset{2}{\psi}_{,a}\\ &+\frac{\kappa^2\omega'(\Psi)}{(2\omega(\Psi)+3)^2}\rho\overset{2}{\psi} + \kappa^2\frac{\omega(\Psi)+2}{2\omega(\Psi)+3}\rho\overset{2}{g}_{00}\\ &-\frac{1}{2}\overset{2}{\psi}_{,a}\overset{2}{g}_{00,a} - \frac{1}{2}\overset{2}{\psi}\triangle\overset{2}{g}_{00} - \overset{2}{\psi}_{,00} - \kappa^2\rho|v|^2\\ &-\kappa^2\frac{\omega(\Psi)+2}{2\omega(\Psi)+3}\rho\Pi - 3\kappa^2\frac{\omega(\Psi)+3}{2\omega(\Psi)+3}p.\end{aligned} \tag{64}$$

As we shall see below, this equation can be solved by the following ansatz:

For consistency, it is useful to check the completeness of this decomposition:

```
In[]:= ans4def = PPN[Met, 4][-LI[0], -LI[0]] == aa[11] * PotentialU[]^2 +
    aa[7] * PotentialPhi1[] + aa[8] * PotentialPhi2[] +
    aa[9] * PotentialPhi3[] + aa[10] * PotentialPhi4[]
```

Out[]= $\overset{4}{g}_{00} = a_7\Phi_1 + a_8\Phi_2 + a_9\Phi_3 + a_{10}\Phi_4 + a_{11}U^2$

```
In[]:= ans4ru = mkrg[ans4def];
```

We then insert this ansatz into the fourth-order field equation, alongside the previously determined solutions at lower velocity order. In order to convert the post-Newtonian potentials to a unified form, from which we can read off the independent equations for the constant coefficients in the ansatz, it is sufficient to replace the divergence terms $V_{a,a}$ and $W_{a,a}$ by time derivatives of U, before replacing all potentials by source terms. This is done as follows:

```
In[]:= eqs4 /. ans4ru /. sol2ru /. sol3ru;
In[]:= Expand[%];
In[]:= ContractMetric[%, OverDerivatives → True, AllowUpperDerivatives → True];
In[]:= PotentialVToU[%];
In[]:= PotentialWToU[%];
In[]:= PotentialToSource[%];
In[]:= ToCanonical[%];
In[]:= SortPDs[%];
In[]:= Expand[%];
In[]:= eqsa4 = Simplify[ScreenDollarIndices[%]];
```

The resulting equation $\overset{4}{\mathcal{E}}_{00}$, which we do not display here for brevity, involves the six independent terms p, $\rho\Pi$, ρU, $U_{,00}$, $\rho|v|^2$, $U_{,a}U_{,a}$, whose constant coefficients must vanish. We thus isolate these constant coefficients:

```
In[]:= eq1 = Simplify[Coefficient[eqsa4, Pressure[]]];
In[]:= eq2 = Simplify[Coefficient[eqsa4, Density[] ∗ InternalEnergy[]]];
In[]:= eq3 = Simplify[Coefficient[eqsa4, Density[] ∗ PotentialU[]]];
In[]:= eq4 = Simplify[Coefficient[eqsa4, ParamD[TimePar, TimePar][PotentialU[]]]];
In[]:= eq5 = Simplify[Coefficient[eqsa4, Density[] ∗ Velocity[−T3a] ∗ Velocity[T3a]]];
In[]:= eq6 = Simplify[Coefficient[eqsa4, PD[−T3a][PotentialU[]] ∗ PD[T3a][PotentialU[]]]];
```

```
In[]:= Simplify[Pressure[] * eq1 + Density[] * InternalEnergy[] * eq2 +
    Density[] * PotentialU[] * eq3 + ParamD[TimePar, TimePar][PotentialU[]] * eq4 +
    Density[] * Velocity[-T3a] * Velocity[T3a] * eq5 +
    PD[-T3a][PotentialU[]] * PD[T3a][PotentialU[]] * eq6 - eqsa4]
Out[]= 0
```

We have thus obtained six equations for the six remaining unknowns $a_0, a_6, \ldots, a_{11}$. It turns out that these equations are linearly independent, so that we can solve for all missing constants:

```
In[]:= sola4 = Simplify[First[Solve[# == 0 & /@ {eq1, eq2, eq3, eq4, eq5, eq6},
    aa /@ Prepend[Range[7, 11], 0]]]];
```

This yields the solution

$$\begin{aligned} a_0 &= \frac{\kappa^2}{4\pi\Psi}\frac{3\omega(\Psi)+4}{2\omega(\Psi)+3}, \quad a_9 = \frac{\kappa^2}{2\pi\Psi}\frac{\omega(\Psi)+2}{2\omega(\Psi)+3},\\ a_8 &= \frac{\kappa^4}{16\pi^2\Psi^2}\frac{12+38\omega(\Psi)+32\omega^2(\Psi)+8\omega^3(\Psi)-\Psi\omega'(\Psi)}{(2\omega(\Psi)+3)^3},\\ a_7 &= \frac{\kappa^2}{2\pi\Psi}, \quad a_{10} = \frac{3\kappa^2}{2\pi\Psi}\frac{\omega(\Psi)+1}{2\omega(\Psi)+3},\\ a_{11} &= -\frac{\kappa^4}{32\pi^2\Psi^2}\frac{48+80\omega(\Psi)+44\omega^2(\Psi)+8\omega^3(\Psi)+\Psi\omega'(\Psi)}{(2\omega(\Psi)+3)^3}. \end{aligned} \tag{65}$$

We check that it is indeed a solution:

```
In[]:= Simplify[eqsa4 /. sola4]
Out[]= 0
```

Since we have now determined a_0, we can complete the partial third-order solution we obtained earlier:

```
In[]:= sol3def = ans3def /. Simplify[sola3 /. sola4];
In[]:= sol3ru = mkrg[sol3def];
```

Using the remaining constants $a_6, \ldots, a_{11}$, we obtain the solution for $\overset{4}{g}_{00}$:

```
In[]:= sol4def = ans4def /. sola4;
In[]:= sol4ru = mkrg[sol4def];
```

Again, we check for consistency that this solves the fourth-order field equation:

```
In[]:= eqs4 /. sol2ru /. sol3ru /. sol4ru;
In[]:= Expand[%];
In[]:= ContractMetric[%, OverDerivatives → True, AllowUpperDerivatives → True];
In[]:= PotentialVToU[%];
In[]:= PotentialWToU[%];
In[]:= PotentialToSource[%];
In[]:= ToCanonical[%];
In[]:= SortPDs[%];
In[]:= Expand[%];
In[]:= Simplify[%]
Out[]= 0
```

6.7 PPN metric and parameters

Using the solution we determined so far, we can now calculate the post-Newtonian metric and parameters. For this purpose, we first isolate the metric components to be determined.

```
In[]:= metcomp = {PPN[Met, 2][-LI[0], -LI[0]], PPN[Met, 2][-T3a, -T3b],
    PPN[Met, 3][-LI[0], -T3a], PPN[Met, 4][-LI[0], -LI[0]]}
```

Out[]= $\left\{\overset{2}{g}_{00}, \overset{2}{g}_{ab}, \overset{3}{g}_{0a}, \overset{4}{g}_{00}\right\}$

We then insert the solution we have found.

```
In[]:= metcomp /. sol2ru /. sol3ru /. sol4ru;
In[]:= ToCanonical[%];
In[]:= Expand[%];
In[]:= ppnmet = Simplify[%];
```

We will compare this to the standard PPN metric.

```
In[]:= stamet = Simplify[MetricToStandard /@ metcomp];
```

This can be done in two steps. First, we choose the normalization constant κ, which corresponds to the effective Newtonian constant, such that $\overset{2}{g}_{00} = 2U$. We define the following equation:

```
In[]:= kappaeq = First[ppnmet] == First[stamet];
```

This takes the form

$$2U = \frac{\kappa^2}{2\pi\Psi}\frac{\omega(\Psi)+2}{2\omega(\Psi)+3}U. \tag{66}$$

To solve this equation, we take its positive root:

```
In[]:= First[Sqrt[FullSimplify[k2 /. Solve[kappaeq /. kappa → Sqrt[k2], k2]]]];
In[]:= kappadef = kappa == %;
In[]:= kapparu = mkrg[kappadef];
```

This yields the solution

$$\kappa = \sqrt{4\pi\Psi\frac{2\omega(\Psi)+3}{\omega(\Psi)+2}}. \tag{67}$$

With this solution, we can now determine the PPN parameters. These are the equations to be solved.

```
In[]:= pareqs = Simplify[ToCanonical[stamet - ppnmet /. kapparu]];
```

To determine the PPN parameters, we isolate the constant coefficients in front of the post-Newtonian potentials $U\delta_{ab}$, V_a, W_a, $\mathcal{A}$, U^2, Φ_W, Φ_1, Φ_2, Φ_3, Φ_4.

```
In[]:= pots = {PotentialU[] BkgMetricS3[-T3a, -T3b], PotentialV[-T3a], PotentialW[-T3a],
    PotentialA[], PotentialU[]^2, PotentialPhiW[],
    PotentialPhi1[], PotentialPhi2[], PotentialPhi3[], PotentialPhi4[]};
In[]:= eqs = DeleteCases[Flatten[Simplify[Outer[Coefficient, pareqs, pots]]], 0];
```

Finally, we can solve for the full set of PPN parameters.

```
In[]:= pars = {ParameterBeta, ParameterGamma, ParameterXi,
    ParameterAlpha1, ParameterAlpha2, ParameterAlpha3,
    ParameterZeta1, ParameterZeta2, ParameterZeta3, ParameterZeta4};
In[]:= parsol = FullSimplify[Solve[# == 0 & /@ eqs, pars][[1]]];
```

This finally yields the solution

$$\gamma = \frac{\omega(\Psi)+1}{\omega(\Psi)+2}, \quad \beta = 1 + \frac{\Psi\omega'(\Psi)}{4(2\omega(\Psi)+3)(\omega(\Psi)+2)^2},$$
$$\alpha_1 = \alpha_2 = \alpha_3 = \zeta_1 = \zeta_2 = \zeta_3 = \zeta_4 = \xi = 0. \tag{68}$$

This is of course the well-known post-Newtonian limit of scalar-tensor gravity with a massless scalar field [31].

7 Summary and outlook

We have presented the Mathematica package *xPPN*, which is based on the tensor algebra suite *xAct* and which implements the PPN formalism in its formulation presented in [5], with extensions to adapt it to the geometries employed in the three formulations of general relativity [26] and modifications thereof. Besides discussing its underlying mathematical concepts and giving a detailed account on its usage, we provided an example application to a simple scalar-tensor theory of gravity. We believe that this package will find wide application to assess the viability of gravity theories by comparing their post-Newtonian limit to observations in the solar system.

Despite the generality of *xPPN*, being applicable to gravity theories based on different geometric foundations, various extensions and modifications are possible and may be implemented in future versions. For example, one may consider the following types of extensions:

1. Alternative formulations of the PPN formalism: A newer approach towards the PPN formalism employs a different density variable [7, Sec. 4], which has the advantage that it simplifies the gauge transformation of the PPN potentials compared to the original formulation. Another approach to simplify the issue of gauge transformations is to resort to a formulation which makes use of gauge-invariant perturbation theory, thereby resolving the necessity of gauge considerations and simplifying the application of the PPN

procedure [32]. Such modifications may be included by changing the expansion of the metric in terms of PPN potentials.

2. Additional post-Newtonian potentials and parameters: Various theories of gravity exhibit a post-Newtonian limit in which the metric perturbation cannot be expressed only in terms of the post-Newtonian potentials used in the standard formalisms, and so more general potentials and corresponding parameters have been introduced. The presence of massive fields leads to the appearance of Yukawa-type potentials [27,28], while higher-order derivatives can be accommodated by further integrals in the post-Newtonian potentials [33,34]. Another class of terms arises from theories which include parity-violating contributions [35–37], or by violation of diffeomorphism invariance [38–40]. Further, one may introduce also fourth-order tensor potentials to expand the term $\overset{4}{g}_{ab}$, which is neglected in the standard PPN formalism, but may be used to describe higher-order corrections to light propagation [13,14]. Such additional potentials may easily be included by defining the corresponding tensor fields and the equations which relate them to the corresponding source terms, in full analogy to the standard PPN potentials.
3. Higher than fourth order in the post-Newtonian expansion: While in the standard PPN formalism implemented in the present version of *xPPN* the metric (and possible other fields present in the theory) are expanded only up to the fourth velocity order, one may also consider higher order terms, and address problems such as the post-Newtonian dynamics of systems involved in the generation of gravitational waves [7,41]. Care must be taken since the Euler equations governing the dynamics of the source matter attain dissipative terms at higher orders, and also the gravitational field exhibits dissipative behavior due to the loss of energy by emitted gravitational waves. Further, such as extension requires the definition of higher-order post-Newtonian potentials, along with corresponding post-Newtonian parameters, whose relation to observations and associated observables must be defined, so that such an extension would be a major task.
4. Generalized post-Newtonian expansion: The standard PPN formalism assumes the validity of a post-Newtonian expansion around a flat Minkowski background. This assumption may be generalized, by allowing for a time-dependent Friedmann–Lemaître–Robertson–Walker background [29], or by including non-perturbative effects arising from the Vainshtein screening mechanism [42].
5. More general geometric frameworks: Another assumption of the standard PPN formalism states that the motion of test masses is governed by a single metric, and so the observable effects on test masses are fully covered by a post-Newtonian expansion of this single metric. This assumption may be relaxed if there are several dynamical metrics present in the considered gravity theory, which lead to more complex test matter dynamics and generalized sources of gravity [43–45]. Also more general connections may be considered, such as a general teleparallel connection [46], as well as Riemann-Cartan [16,33,34,47,48] or metric-affine geometry [49]. In theories of this class additional matter currents may appear from a coupling of spin to gravity, which gives rise to additional source terms, which must be accommodated for by further PPN potentials, together with their respective equations of motion and conservation laws generalizing the Euler equations [50–57].

Acknowledgements The author thanks Yuri Obukhov and Dirk Puetzfeld for helpful comments on the manuscript and pointing out related work. He gratefully acknowledges the full support by the Estonian Research Council through the Personal Research Funding project PRG356, as well as the European Regional Development Fund through the Center of Excellence TK133 “The Dark Side of the Universe”.

Data Availability Statement This manuscript has no associated data or the data will not be deposited. [Authors’ comment: This article is accompanied by a software package available from Ref. [9]. The latter is continuously developed and extended, and it is recommended to always use the most up to date version in order to use all implemented features. Hence, no snapshot version is deposited, as it would soon be outdated.]

Funded by SCOAP^3.

Appendix A: Implementation notes

Although *xTensor* offers a possibility to split tensor indices for product manifolds and to project tensors to submanifolds, while *xPert* provides an implementation of a perturbative expansion of tensor fields, these approaches are not the most well-adapted to the mathematical concepts discussed in Sect. 3. While it may still be possible to use these packages for the task at hand, *xPPN* provides its own implementations of these concepts, in order to match as closely as possible with the PPN formalism, and to simplify the different perturbative treatment of space and time components of tensors and in particular their derivatives. This appendix contains a few notes how the functions detailed in the main part of this article are implemented. The internal representation of split tensor fields is shown in Sect. 1. In Sect. 1, we give an overview of the algorithm used to obtain this decomposition for arbitrary tensor fields. Finally, in Sect. 1 we give a few notes on the implementation of the perturbative expansion in velocity orders. The aim of this technical appendix is to give an insight to the internal data structures and algorithms used by *xPPN*, which is not necessary for its use, but may be helpful for readers who intend to develop extensions to *xPPN* or similar packages based on *xAct* in other areas of physics.

A.1 Internal representation of decomposed tensors

As detailed in Sect. 3.1, we adhere to the following mathematical interpretation of the $3+1$ split of tensor fields, which turns out to be convenient for the PPN formalism:

1. The spacetime manifold is regarded as a product manifold $M_4 \cong T_1 \times S_3$.
2. The components of any tensor defined on the spacetime manifold M_4 with respect to adapted coordinates (t, x^a) are split into time and space components. For example, given a vector A^α, one has a decomposition into a time component A^0 and spatial components A^a.
3. The decomposed parts of a tensor on M_4 are regarded as tensors on S_3, obtained as a time slice in a foliation of spacetime, carrying an additional dependence on time t, where time is treated not as a coordinate, but as a parameter.
4. The decomposition simplifies in the presence of tensor symmetries. For example, given an antisymmetric tensor $A_{\alpha\beta} = A_{[\alpha\beta]}$, the component A_{00} vanishes, indices in A_{a0} may equivalently be sorted in canonical (lexicographic) order to yield $-A_{0a}$ and A_{ab} is a tensor which inherits the antisymmetry in its two indices.

The two mentioned approaches implemented in *xTensor* do not match these criteria:

1. The projection approach using normal vector fields and induced metrics does not perform any decomposition of indices; the equivalent of the time component A^0 retains its tensor rank, and is hence represented by a vector, although being projected onto (and hence tangent to) a one-dimensional manifold. One may obtain a scalar by contracting with the normal vector field, at the cost of computational complexity for carrying this contracted factor through all calculations.[3]
2. The split along product manifolds also retains the tensor rank. Instead of the inert time index 0 and thus being treated as scalar, time components carry a coordinate index associated to the time manifold, which therefore must satisfy the rules imposed by *xTensor* on such indices (such as the restriction on the number of occurrences, uniqueness in expressions, summation over dummy indices etc.).

Furthermore, in both approaches time retains its nature of a coordinate. Therefore, *xPPN* uses the following approach to decomposing tensor fields instead. The components of the $3+1$ split of tensors are represented by the inert function `PPNTensor` in *xPPN*, which resides in the private context `xAct`xPPN`Private``. It may be used in the following two forms:

1. `PPNTensor[`h`,{`s_1`,`s_2`,...,`s_k`}]`, where h is a valid tensor head belonging to a tensor on the spacetime manifold M_4 and $s_1, s_2, \ldots, s_k$ is a list of index slots, represents a component in the $3+1$ decomposition of the tensor with head h. The particular component is selected by the list of tensor slots; see the list below for valid values and their interpretation.
2. `PPNTensor[`h`,{`s_1`,`s_2`,...,`s_k`},`n`]`, where n is a non-negative integer and $h, s_1, s_2, \ldots, s_k$ are as above, represents the term of velocity order $\mathcal{O}(n)$ in the perturbative expansion of the aforementioned component represented by `PPNTensor[`h`, {`s_1`,`s_2`,...,`s_k`}]`.

The valid values for the slots $s_1, s_2, \ldots, s_k$ depends on the slots $S_1, S_2, \ldots, S_k$ of h. The number k of slots given (i.e., the length of the list) must match the number of slots of h. Further, for every slot S_i of h the following rules must be satisfied:

1. The character of the slots must match, i.e., if S_i is an upper (lower) index, then also s_i must be an upper (lower) index.
2. If S_i is ±`TangentMfSpacetime`, then s_i must be one of ±`Labels` or ±`TangentMfSpace`.
3. If S_i is ±`LorentzMfSpacetime`, then s_i must be one of ±`Labels` or ±`LorentzMfSpace`.
4. If S_i is any other slot and hence does not belong to a vector bundle over M_4, then s_i must be the same as S_i.

The interpretation should be evident: every tangent or Lorentz index associated to the spacetime manifold M_4 is replaced either by a corresponding index on the space manifold or the special inert value `LI[0]`, belonging to `Labels`, which represents the time component. *xPPN* performs the decomposition automatically whenever a tensor field is defined.

[3] This approach is used in *xPand* [11], and such contractions are then replaced by tensors of lower rank which are pre-defined for the space and time components of metric perturbations. Here we aim to avoid manually defining such a decomposition, in favor of an algorithm which

defines upvalues for the following *xTensor* functions applied to `PPNTensor` objects obeying any of the two forms listed above:

1. `xTensorQ` yields `True` if h is a valid tensor head on the spacetime manifold and the slots satisfy the conditions listed above, and `False` otherwise.
2. `SlotsOfTensor` returns the list $\{s_1, s_2, \ldots, s_k\}$ of tensor slots.
3. `DependenciesOfTensor` gives the dependencies of h, but with `MfSpacetime` replaced by `MfSpace` and `TimePar`. If the tensor depends on any other parameters or manifolds, then these dependencies are preserved.
4. `SymmetryGroupOfTensor` returns the remaining symmetry group of the $3+1$ decomposed tensor.
5. `PrintAs` yields the same symbol `PrintAs[`h`]` which is used for printing h if no velocity order is given, and otherwise adds the velocity order n as an overscript $\overset{n}{h}$.

Further, `PPNTensor` automatically applies rules which are obtained from index symmetries. These are calculated whenever a tensor field is defined, as explained below.

A.2 $3+1$ spacetime split algorithm

In order to retain only independent components in the $3+1$ split of a tensor, *xPPN* examines tensor symmetries whenever a new tensor on the spacetime manifold is defined. It does so by extending the *xTensor* function `DefTensor`, using the extensibility framework implemented in *xAct*, to perform the following steps:

1. Determine which index slots of the tensor are associated to the spacetime manifold. If a tensor carries additional, internal indices, then they will be treated separately and ignored in the following steps.
2. Calculate the symmetry group acting on the spacetime indices only, i.e., factor out any possible symmetries acting on internal indices and also ignore them in the following steps.
3. Create a list of all possible ways to split the spacetime indices into space and time components.
4. Consider the action of the symmetry group on the splits from the previous step and determine the orbits of this action. For example, if a tensor carries a symmetry (or antisymmetry) in two indices, then there is no difference between choosing the first index to be temporal and the second index spatial or vice versa, and so these two possible splits belong to the same orbit; these two possible ways to arrange the indices of the split tensor are equivalent.
5. For each orbit, perform the following steps:
   (a) Pick a representative, i.e., from all equivalent ways to arrange the indices, choose the one which comes first in canonical (lexicographic) order.
   (b) Check whether any of the elements of the subgroup which permutes only the temporal indices comes with an antisymmetry, i.e., changes the sign of the tensor. If this is the case, this component must vanish identically, since permuting only temporal indices must leave the tensor invariant, and so it must be equal to its negative. In this case, define all index combinations in this orbit to be an alias for the zero tensor `Zero`.
   (c) If the chosen representative does not vanish, conclude with the following steps:
      (i) Determine the remaining symmetry group acting on the spatial indices only.
      (ii) Define a tensor on a purely spatial manifold, whose temporal and spatial indices match with the representative, which carries the symmetry determined in the previous step in the spacetime indices and any inherit symmetry in internal indices, and which in addition depends on a time parameter.
      (iii) Define all other arrangements of temporal and spatial indices which are equivalent to the representative as alias for the previously defined tensor or its negative, taking into account possible sign changes when indices are permuted.

To illustrate these steps, consider the case of defining an antisymmetric tensor $A_{\alpha\beta} = A_{[\alpha\beta]}$ on the spacetime manifold, by invoking

```
In[]:= DefTensor[A[-T4α, -T4β], MfSpacetime, Antisymmetric[{1, 2}]]
```

Then the following steps are carried out automatically by *xPPN*:

1. There are no internal indices, and so the spacetime related symmetry operations acting on the tensor indices are the identity and the swapping of the two indices, where the latter also changes the sign of the tensor.
2. Each of the two indices of $A_{\alpha\beta}$ splits into time and space. Hence, there are four possible decomposed tensor fields: A_{00}, A_{0a}, A_{a0}, A_{ab}.
3. The two components A_{0a} and A_{a0} are transformed into each other under the index symmetry of the original tensor fields; hence, they belong to the same orbit of the action of the symmetry group. The remaining components are the sole elements in their respective orbits.
4. The three orbits are examined separately:

   (a) For A_{00} one finds that the swap operation acts on temporal indices only, but also changes the sign of the tensor. Hence, $A_{00} = -A_{00} = 0$ and the tensor is defined as an alias of `Zero`:

   ```
   A /: PPNTensor[A, {−Labels, −Labels}] := Zero;
   A /: PPNTensor[A, {−Labels, −Labels}, _] := Zero;
   ```

   (b) For A_{0a} and A_{a0} the former arrangement of indices is chosen as a representative, since the indices are in canonical (lexicographic) order. A purely spatial tensor A_{0a} is defined, which has one free index slot. A_{a0} is defined as an alias for $-A_{0a}$:

   ```
   A /: PPNTensor[A, {−TangentMfSpace, −Labels}] :=
       (−PPNTensor[A, {−Labels, −TangentMfSpace}][#2, #1] &);
   ```

   (c) Another spatial tensor A_{ab} is defined, which is antisymmetric in its two index slots. The tensor symmetry is again associated to the symbol `A`.

Note that all properties of these tensors, such as their symmetries and tensor slots, are associated with the symbol (tensor head) which is used in the original call to `DefTensor` to define the tensor on the spacetime manifold, and no further symbols are introduced.

A.3 Velocity order decomposition algorithm

In order to implement the rules for the perturbative expansion in velocity orders detailed in Sect. 3.2, a few special cases have been defined for the function `VelocityOrder` shown in Sect. 5.4. For the product rule (29), it is most convenient to rely on Mathematica's pattern matching and recursive function application functionality. Given a product $A = A_1 \cdots A_N$ of N tensors, one may split off the first factor,

$$A = A_1 \tilde{A}, \quad \tilde{A} = A_2 \cdots A_N, \tag{A.1}$$

and then recursively apply the rule

$$\overset{n}{A} = \sum_{k=0}^{n} \overset{k}{A}_1 \overset{n-k}{\tilde{A}}. \tag{A.2}$$

This is repeated until only a single factor is left. The implementation then takes the following simple form.

```
VelocityOrder[Times[ex0_, ex1__], n_, opt : OptionsPattern[]] :=
    Module[{k}, Sum[VelocityOrder[ex0, k, opt] *
        VelocityOrder[Times[ex1], n - k, opt], {k, 0, n}]];
```

Next, we come to the relation (31) for the perturbative expansion of expressions which are given as functions of an arbitrary number of arguments. To evaluate this formula, it is useful to define the formal series

$$\tilde{A}_i(\epsilon) = \sum_{k=0}^{\infty} \epsilon^k \overset{k}{A}_i \tag{A.3}$$

for the tensor fields $A_1, \ldots, A_N$. Observe that the n'th order series coefficient

$$\frac{1}{n!} \frac{\mathrm{d}^n}{\mathrm{d}\epsilon^n} f(\tilde{A}_1(\epsilon), \ldots, \tilde{A}_N(\epsilon))\bigg|_{\epsilon=0} \\ = \sum_{p_{ik}} f^{(p_{11}+\ldots+p_{1n},\ldots,p_{N1}+\ldots+p_{Nn})}\left(\overset{0}{A}_1, \ldots, \overset{0}{A}_N\right) \prod_{i=1}^{N} \prod_{k=1}^{n} \frac{\overset{k}{A}_i^{p_{ik}}}{p_{ik}!}, \tag{A.4}$$

where the sum runs over

$$p_{ik} \in \{0, \ldots, q_k\}, \quad \sum_{i=1}^{N} p_{ik} = q_k \,; \quad i \in \{1, \ldots, N\} \tag{A.5}$$

and

$$q_k \in \{1, \ldots, n\}, \quad \sum_{k=1}^{n} k q_k = n, \tag{A.6}$$

is exactly the term of n'th velocity order in the expansion (31). Hence, it can be obtained as follows.

```
VelocityOrder[fkt_ ? ScalarFunctionQ[args__], n_, opt : OptionsPattern[]] :=
    Module[{t, i}, SeriesCoefficient[
        fkt @@ (Sum[t^i * VelocityOrder[#, i, opt], {i, 0, n}] & /@ {args}),
    {t, 0, n}]];
```

Finally, we have encountered the rule (32) that time derivatives are weighted with an additional velocity order. Since time derivatives are represented by parameter derivs with respect to `TimePar`, this behavior is achieved by counting time derivatives as follows.

```
VelocityOrder[ParamD[t : TimePar ..][expr_], n_, opt : OptionsPattern[]] :=
    ParamD[t][VelocityOrder[expr, n - Length[{t}], opt]];
```

Note that partial derivatives with respect to spatial coordinates do not incur additional velocity orders.

```
VelocityOrder[PD[i_][expr_], n_, opt : OptionsPattern[]] :=
    PD[i][VelocityOrder[expr, n, opt]];
```

References

1. K. Nordtvedt, Equivalence principle for massive bodies. 2. Theory. Phys. Rev. **169**, 1017–1025 (1968). https://doi.org/10.1103/PhysRev.169.1017
2. K.S. Thorne, C.M. Will, Theoretical frameworks for testing relativistic gravity. 1. Foundations. Astrophys. J. **163**, 595–610 (1971). https://doi.org/10.1086/150803
3. C.M. Will, Theoretical frameworks for testing relativistic gravity. 2. Parametrized post-Newtonian hydrodynamics, and the Nordtvedt effect. Astrophys. J. **163**, 611–627 (1971). https://doi.org/10.1086/150804
4. C.M. Will, Theoretical frameworks for testing relativistic gravity. 3. Conservation laws, Lorentz invariance and values of the PPN parameters. Astrophys. J. **169**, 125–140 (1971). https://doi.org/10.1086/151124
5. C.M. Will, *Theory and Experiment in Gravitational Physics* (Cambridge University Press, Cambridge, 1993). https://doi.org/10.1017/CBO9780511564246
6. C.M. Will, The confrontation between general relativity and experiment. Living Rev. Relativ. **17**, 4 (2014). https://doi.org/10.12942/lrr-2014-4. arXiv:1403.7377 [gr-qc]
7. C.M. Will, *Theory and Experiment in Gravitational Physics* (Cambridge University Press, 2018). https://www.cambridge.org/academic/subjects/physics/cosmology-relativity-and-gravitation/theory-and-experiment-gravitational-physics-2nd-edition?format=AR&isbn=9781108679824
8. D. Puetzfeld, PROCRUSTES: a computer algebra package for post-Newtonian calculations in general relativity. Comput. Phys. Commun. **175**, 497–508 (2006). https://doi.org/10.1016/J.CPC.2006.07.003. arXiv:gr-qc/0610081
9. M. Hohmann, xPPN (2020). http://geomgrav.fi.ut.ee/software/xPPN.html
10. J.M. Martín-García, xAct: efficient tensor computer algebra for the Wolfram Language (2002). http://www.xact.es/
11. C. Pitrou, X. Roy, O. Umeh, xPand: an algorithm for perturbing homogeneous cosmologies. Class. Quantum Gravity **30**, 165002 (2013). https://doi.org/10.1088/0264-9381/30/16/165002. arXiv:1302.6174 [astro-ph.CO]
12. D. Brizuela, J.M. Martin-Garcia, G.A. MenaMarugan, xPert: computer algebra for metric perturbation theory. Gen. Relativ. Gravit. **41**, 2415–2431 (2009). https://doi.org/10.1007/s10714-009-0773-2. arXiv:0807.0824 [gr-qc]
13. G.W. Richter, R.A. Matzner, Second-order contributions to gravitational deflection of light in the parametrized post-Newtonian formalism. Phys. Rev. D **26**, 1219–1224 (1982). https://doi.org/10.1103/PhysRevD.26.1219
14. G.W. Richter, R.A. Matzner, Second-order contributions to gravitational deflection of light in the parametrized post-Newtonian formalism. II. Photon orbits and deflections in three dimensions. Phys. Rev. D **26**, 2549–2556 (1982). https://doi.org/10.1103/PhysRevD.26.2549
15. J. Nitsch, F.W. Hehl, Translational Gauge theory of gravity: Post-newtonian approximation and spin precession. Phys. Lett. **90B**, 98–102 (1980). https://doi.org/10.1016/0370-2693(80)90059-3
16. L.L. Smalley, Postnewtonian approximation of the Poincare gauge theory of gravitation. Phys. Rev. D **21**, 328–331 (1980). https://doi.org/10.1103/PhysRevD.21.328
17. J. Hayward, Scalar tetrad theories of gravity. Gen. Relativ. Gravit. **13**, 43–55 (1981). https://doi.org/10.1007/BF00766297
18. U. Ualikhanova, M. Hohmann, Parameterized post-Newtonian limit of general teleparallel gravity theories. Phys. Rev. D **100**, 104011 (2019). https://doi.org/10.1103/PhysRevD.100.104011. arXiv:1907.08178 [gr-qc]
19. M. Hohmann, General cosmological perturbations in teleparallel gravity (2020). arXiv:2011.02491 [gr-qc]
20. R. Aldrovandi, J.G. Pereira, *Teleparallel Gravity*, vol. 173 (Springer, Dordrecht, 2013). https://doi.org/10.1007/978-94-007-5143-9
21. M. Krssak, R.J. Van Den Hoogen, J.G. Pereira, C.G. Boehmer, A.A. Coley, Teleparallel theories of gravity: illuminating a fully invariant approach. Class. Quantum Gravity **36**, 183001 (2019). https://doi.org/10.1088/1361-6382/ab2e1f. arXiv:1810.12932 [gr-qc]
22. A. Golovnev, T. Koivisto, M. Sandstad, On the covariance of teleparallel gravity theories. Class. Quantum Gravity **34**, 145013 (2017). https://doi.org/10.1088/1361-6382/aa7830. arXiv:1701.06271 [gr-qc]
23. J.M. Nester, H.-J. Yo, Symmetric teleparallel general relativity. Chin. J. Phys. **37**, 113 (1999). arXiv:gr-qc/9809049
24. J.B. Jiménez, L. Heisenberg, T. Koivisto, Coincident general relativity. Phys. Rev. D **98**, 044048 (2018). https://doi.org/10.1103/PhysRevD.98.044048. arXiv:1710.03116 [gr-qc]
25. K. Flathmann, M. Hohmann, Post-Newtonian limit of generalized symmetric teleparallel gravity (2020). arXiv:2012.12875 [gr-qc]
26. J.B. Jiménez, L. Heisenberg, T.S. Koivisto, The geometrical trinity of gravity. Universe **5**, 173 (2019). https://doi.org/10.3390/universe5070173. arXiv:1903.06830 [hep-th]
27. H.W. Zaglauer, *Phenomenological aspects of scalar fields in astrophysics, cosmology and particle physics*. Ph.D. thesis. Washington U., St. Louis (1990). http://wwwlib.umi.com/dissertations/fullcit?p9103178
28. T. Helbig, Gravitational effects of light scalar particles. Astrophys. J. **382**, 223–232 (1991). https://doi.org/10.1086/170710
29. V.A.A. Sanghai, T. Clifton, Parameterized post-Newtonian cosmology. Class. Quantum Gravity **34**, 065003 (2017). https://doi.org/10.1088/1361-6382/aa5d75. arXiv:1610.08039 [gr-qc]
30. T. Nutma, xTras: a field-theory inspired xAct package for Mathematica. Comput. Phys. Commun. **185**, 1719–1738 (2014). https://doi.org/10.1016/j.cpc.2014.02.006. arXiv:1308.3493 [cs.SC]
31. K. Nordtvedt, Jr., Post-Newtonian metric for a general class of scalar tensor gravitational theories and observational consequences. Astrophys. J. **161**, 1059–1067 (1970). https://doi.org/10.1086/150607
32. M. Hohmann, Gauge-invariant approach to the parametrized post-Newtonian formalism. Phys. Rev. D **101**, 024061 (2020). https://doi.org/10.1103/PhysRevD.101.024061. arXiv:1910.09245 [gr-qc]
33. M.S. Gladchenko, V.N. Ponomarev, V.V. Zhytnikov, PPN metric and PPN torsion in the quadratic Poincare gauge theory of gravity. Phys. Lett. B **241**, 67–69 (1990). https://doi.org/10.1016/0370-2693(90)91488-W

34. M.S. Gladchenko, V.V. Zhytnikov, PostNewtonian effects in the quadratic Poincare gauge theory of gravitation. Phys. Rev. D **50**, 5060–5071 (1994). https://doi.org/10.1103/PhysRevD.50.5060
35. S. Alexander, N. Yunes, A New PPN parameter to test Chern–Simons gravity. Phys. Rev. Lett. **99**, 241101 (2007). https://doi.org/10.1103/PhysRevLett.99.241101. arXiv:hep-th/0703265
36. S. Alexander, N. Yunes, Parametrized post-Newtonian expansion of Chern-Simons gravity. Phys. Rev. D **75**, 124022 (2007). https://doi.org/10.1103/PhysRevD.75.124022. arXiv:0704.0299 [hep-th]
37. S. Alexander, N. Yunes, Chern–Simons modified general relativity. Phys. Rep. **480**, 1–55 (2009). https://doi.org/10.1016/j.physrep.2009.07.002. arXiv:0907.2562 [hep-th]
38. K. Lin, S. Mukohyama, A. Wang, Solar system tests and interpretation of gauge field and Newtonian prepotential in general covariant Hořava-Lifshitz gravity. Phys. Rev. D **86**, 104024 (2012). https://doi.org/10.1103/PhysRevD.86.104024. arXiv:1206.1338 [hep-th]
39. K. Lin, A. Wang, Static post-Newtonian limits in nonprojectable Hořava-Lifshitz gravity with an extra U(1) symmetry. Phys. Rev. D **87**, 084041 (2013). https://doi.org/10.1103/PhysRevD.87.084041. arXiv:1212.6794 [hep-th]
40. K. Lin, S. Mukohyama, A. Wang, T. Zhu, Post-Newtonian approximations in the Hořava-Lifshitz gravity with extra U(1) symmetry. Phys. Rev. D **89**, 084022 (2014). https://doi.org/10.1103/PhysRevD.89.084022. arXiv:1310.6666 [hep-ph]
41. L. Blanchet, Gravitational radiation from post-Newtonian sources and inspiralling compact binaries. Living Rev. Relativ. **17**, 2 (2014). https://doi.org/10.12942/lrr-2014-2. arXiv:1310.1528 [gr-qc]
42. A. Avilez-Lopez, A. Padilla, P.M. Saffin, C. Skordis, The parametrized post-Newtonian-Vainshteinian formalism. JCAP **1506**, 044 (2015). https://doi.org/10.1088/1475-7516/2015/06/044. arXiv:1501.01985 [gr-qc]
43. T. Clifton, M. Banados, C. Skordis, The parameterised post-Newtonian limit of bimetric theories of gravity. Class. Quantum Gravity **27**, 235020 (2010). https://doi.org/10.1088/0264-9381/27/23/235020. arXiv:1006.5619 [gr-qc]
44. M. Hohmann, M.N.R. Wohlfarth, Multimetric extension of the PPN formalism: experimental consistency of repulsive gravity. Phys. Rev. D **82**, 084028 (2010). https://doi.org/10.1103/PhysRevD.82.084028. arXiv:1007.4945 [gr-qc]
45. M. Hohmann, Parameterized post-Newtonian formalism for multimetric gravity. Class. Quantum Gravity **31**, 135003 (2014). https://doi.org/10.1088/0264-9381/31/13/135003. arXiv:1309.7787 [gr-qc]
46. J. BeltránJiménez, L. Heisenberg, D. Iosifidis, A. Jiménez-Cano, T.S. Koivisto, General teleparallel quadratic gravity. Phys. Lett. B **805**, 135422 (2020). https://doi.org/10.1016/j.physletb.2020.135422. arXiv:1909.09045 [gr-qc]
47. M. Castagnino, M.L. Levinas, N. Umerez, On the post-Newtonian approximation of the Einstein–Cartan–Sciama–Kibble theory. Gen. Relativ. Gravit. **17**, 683 (1985). https://doi.org/10.1007/BF00763029
48. M. Castagnino, M.L. Levinas, On the post-Newtonian approximation of the ECSK theory II. Gen. Relativ. Gravit. **19**, 545 (1987). https://doi.org/10.1007/BF00762553
49. F.W. Hehl, J.D. McCrea, E.W. Mielke, Y. Ne'eman, Metric affine gauge theory of gravity field equations, Noether identities, world spinors, and breaking of dilation invariance. Phys. Rep. **258**, 1–171 (1995). https://doi.org/10.1016/0370-1573(94)00111-F. arXiv:gr-qc/9402012
50. J. Weyssenhoff, A. Raabe, Relativistic dynamics of spin-fluids and spin-particles. Acta Phys. Polon. **9**, 7–18 (1947)
51. J.R. Ray, L.L. Smalley, Spinning fluids in the Einstein–Cartan theory. Phys. Rev. D **27**, 1383 (1983). https://doi.org/10.1103/PhysRevD.27.1383
52. W. Kopczynski, Lagrangian dynamics of particles and fluids with intrinsic spin in Einstein–Cartan space-time. Phys. Rev. D **34**, 352–356 (1986). https://doi.org/10.1103/PhysRevD.34.352
53. Y.N. Obukhov, V.A. Korotkii, The Weyssenhoff fluid in Einstein–Cartan theory. Class. Quantum Gravity **4**, 1633–1657 (1987). https://doi.org/10.1088/0264-9381/4/6/021
54. Y.N. Obukhov, R. Tresguerres, Hyperfluid: a model of classical matter with hypermomentum. Phys. Lett. A **184**, 17–22 (1993)
55. Y.N. Obukhov, On a model of an unconstrained hyperfluid. Phys. Lett. A **210**, 163–167 (1996). https://doi.org/10.1016/S0375-9601(96)80004-1. arXiv:gr-qc/0008014
56. D. Puetzfeld, Y.N. Obukhov, Equations of motion in metric-affine gravity: a covariant unified framework. Phys. Rev. D **90**, 084034 (2014). https://doi.org/10.1103/PhysRevD.90.084034. arXiv:1408.5669 [gr-qc]
57. D. Iosifidis, Cosmological hyperfluids, torsion and non-metricity. Eur. Phys. J. C **80**, 1042 (2020). https://doi.org/10.1140/epjc/s10052-020-08634-z. arXiv:2003.07384 [gr-qc]