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

https://github.com/xenos1984/xPPN

Manuel Hohmann

Laboratory of Theoretical Physics - Institute of Physics - University of Tartu
Center of Excellence "The Dark Side of the Universe"

European Union
European Regional
Development Fund

Investing
in your future

Space Science @ Drop Tower Seminar - 2. September 2020

Motivation

- Parametrized post-Newtonian formalism:
    - Weak-field approximation of metric gravity theories.
    - Assumes particular coordinate system (“universe rest frame”).
    - Characterizes gravity theories by 10 (constant) parameters.
    - Parameters closely related to solar system observations.

Motivation

- Parametrized post-Newtonian formalism:
    - Weak-field approximation of metric gravity theories.
    - Assumes particular coordinate system (“universe rest frame”).
    - Characterizes gravity theories by 10 (constant) parameters.
    - Parameters closely related to solar system observations.
- Properties of the PPN formalism:
    - ✓ Need to solve only linear equations at each perturbation order.
    - ↯ Equations may be lengthy, coupled and difficult to disentangle.
    - ↯ Numerous relations and transformation rules needed to solve equations.

Motivation

- Parametrized post-Newtonian formalism:
  - Weak-field approximation of metric gravity theories.
  - Assumes particular coordinate system (“universe rest frame”).
  - Characterizes gravity theories by 10 (constant) parameters.
  - Parameters closely related to solar system observations.
- Properties of the PPN formalism:
  - ✓ Need to solve only linear equations at each perturbation order.
  - ↯ Equations may be lengthy, coupled and difficult to disentangle.
  - ↯ Numerous relations and transformation rules needed to solve equations.

⇝ Implement generic PPN formalism using computer tensor algebra.

Motivation

- Parametrized post-Newtonian formalism:
  - ∘ Weak-field approximation of metric gravity theories.
  - ∘ Assumes particular coordinate system (“universe rest frame”).
  - ∘ Characterizes gravity theories by 10 (constant) parameters.
  - ∘ Parameters closely related to solar system observations.
- Properties of the PPN formalism:
  - ✓ Need to solve only linear equations at each perturbation order.
  - ↯ Equations may be lengthy, coupled and difficult to disentangle.
  - ↯ Numerous relations and transformation rules needed to solve equations.

⇝ Implement generic PPN formalism using computer tensor algebra.

- Implementation as package using *xAct* for Mathematica:
  - ∘ Mathematica offers powerful routines for symbolic calculations.
  - ∘ *xAct* implements numerous functions for tensor algebra.
  - ∘ *xAct* can easily be extended with new functionality.

Outline

Outline

1. Parametrized post-Newtonian formalism
2. *xPPN*: an implementation of the PPN formalism
3. Example: PPN limit of scalar-tensor gravity
4. Conclusion

Post-Newtonian matter and velocity orders

- Energy-momentum tensor of a perfect fluid:

$$\Theta^{\mu\nu} = (\rho + \rho\Pi + p)u^{\mu}u^{\nu} + pg^{\mu\nu}\,.$$

  - Rest mass density ρ.
  - Specific internal energy Π.
  - Pressure p.
  - Four-velocity u^{μ}.

Post-Newtonian matter and velocity orders

- Energy-momentum tensor of a perfect fluid:

$$\Theta^{\mu\nu} = (\rho + \rho\Pi + p)u^{\mu}u^{\nu} + pg^{\mu\nu}\,.$$

  - Rest mass density ρ.
  - Specific internal energy Π.
  - Pressure p.
  - Four-velocity u^{μ}.
- Universe rest frame and slow-moving source matter:
  - Velocity of the source matter: $v^i = u^i/u^0$.
  - Assume that source matter is slow-moving: $|\vec{v}| \ll 1$.
  - Use $\epsilon = |\vec{v}|$ as perturbation parameter.

Post-Newtonian matter and velocity orders

- Energy-momentum tensor of a perfect fluid:

$$\Theta^{\mu\nu} = (\rho + \rho\Pi + p)u^{\mu}u^{\nu} + pg^{\mu\nu}\,.$$

  - Rest mass density $\rho \sim \mathcal{O}(2)$.
  - Specific internal energy $\Pi \sim \mathcal{O}(2)$.
  - Pressure $p \sim \mathcal{O}(4)$.
  - Four-velocity u^{μ}.
- Universe rest frame and slow-moving source matter:
  - Velocity of the source matter: $v^i = u^i/u^0$.
  - Assume that source matter is slow-moving: $|\vec{v}| \ll 1$.
  - Use $\epsilon = |\vec{v}|$ as perturbation parameter.
- Assign velocity orders $\mathcal{O}(n) \sim \epsilon^n$ to all quantities based on solar system.

Post-Newtonian matter and velocity orders

- Energy-momentum tensor of a perfect fluid:

$$\Theta^{\mu\nu} = (\rho + \rho\Pi + p)u^{\mu}u^{\nu} + pg^{\mu\nu}\,.$$

  - Rest mass density $\rho \sim \mathcal{O}(2)$.
  - Specific internal energy $\Pi \sim \mathcal{O}(2)$.
  - Pressure $p \sim \mathcal{O}(4)$.
  - Four-velocity u^{μ}.
- Universe rest frame and slow-moving source matter:
  - Velocity of the source matter: $v^i = u^i/u^0$.
  - Assume that source matter is slow-moving: $|\vec{v}| \ll 1$.
  - Use $\epsilon = |\vec{v}|$ as perturbation parameter.
- Assign velocity orders $\mathcal{O}(n) \sim \epsilon^n$ to all quantities based on solar system.
- Quasi-static: assign additional $\mathcal{O}(1)$ to time derivatives ∂_0.

Post-Newtonian metric

- Standard post-Newtonian metric expansion:

$$g_{\mu\nu} = \overset{0}{g}_{\mu\nu} + \overset{1}{g}_{\mu\nu} + \overset{2}{g}_{\mu\nu} + \overset{3}{g}_{\mu\nu} + \overset{4}{g}_{\mu\nu} + \mathcal{O}(5)\,.$$

Post-Newtonian metric

- Standard post-Newtonian metric expansion:

$$g_{\mu\nu} = \overset{0}{g}_{\mu\nu} + \overset{1}{g}_{\mu\nu} + \overset{2}{g}_{\mu\nu} + \overset{3}{g}_{\mu\nu} + \overset{4}{g}_{\mu\nu} + \mathcal{O}(5)\,.$$

- Background metric given by Minkowski metric: $\overset{0}{g}_{\mu\nu} = \eta_{\mu\nu}$.

Post-Newtonian metric

- Standard post-Newtonian metric expansion:

$$g_{\mu\nu} = \overset{0}{g}_{\mu\nu} + \overset{1}{g}_{\mu\nu} + \overset{2}{g}_{\mu\nu} + \overset{3}{g}_{\mu\nu} + \overset{4}{g}_{\mu\nu} + \mathcal{O}(5)\,.$$

- Background metric given by Minkowski metric: $\overset{0}{g}_{\mu\nu} = \eta_{\mu\nu}$.
- Only terms up to fourth velocity order $\mathcal{O}(4)$ are considered.

Post-Newtonian metric

- Standard post-Newtonian metric expansion:

$$g_{\mu\nu} = \overset{0}{g}_{\mu\nu} + \overset{1}{g}_{\mu\nu} + \overset{2}{g}_{\mu\nu} + \overset{3}{g}_{\mu\nu} + \overset{4}{g}_{\mu\nu} + \mathcal{O}(5)\,.$$

- Background metric given by Minkowski metric: $\overset{0}{g}_{\mu\nu} = \eta_{\mu\nu}$.
- Only terms up to fourth velocity order $\mathcal{O}(4)$ are considered.
- Only certain components are relevant and non-vanishing:

$$\overset{2}{g}_{00}\,, \quad \overset{2}{g}_{ij}\,, \quad \overset{3}{g}_{0i}\,, \quad \overset{4}{g}_{00}\,, \quad \overset{4}{g}_{ij}\,.$$

Post-Newtonian metric

- Standard post-Newtonian metric expansion:

$$g_{\mu\nu} = \overset{0}{g}_{\mu\nu} + \overset{1}{g}_{\mu\nu} + \overset{2}{g}_{\mu\nu} + \overset{3}{g}_{\mu\nu} + \overset{4}{g}_{\mu\nu} + \mathcal{O}(5)\,.$$

- Background metric given by Minkowski metric: $\overset{0}{g}_{\mu\nu} = \eta_{\mu\nu}$.
- Only terms up to fourth velocity order $\mathcal{O}(4)$ are considered.
- Only certain components are relevant and non-vanishing:

$$\overset{2}{g}_{00}\,, \quad \overset{2}{g}_{ij}\,, \quad \overset{3}{g}_{0i}\,, \quad \overset{4}{g}_{00}\,, \quad \overset{4}{g}_{ij}\,.$$

- $\overset{4}{g}_{ij}$ not used in standard PPN formalism, but may couple to other components.

Standard post-Newtonian gauge

- PPN formalism assumes fixed standard gauge.

Standard post-Newtonian gauge

- PPN formalism assumes fixed standard gauge.
- Metric in standard PPN gauge:

$$\begin{aligned}
\overset{2}{g}_{00} &= 2U\,, \\
\overset{2}{g}_{ij} &= 2\gamma U\delta_{ij}\,, \\
\overset{3}{g}_{0i} &= -\frac{1}{2}(3+4\gamma+\alpha_1-\alpha_2+\zeta_1-2\xi)V_i-\frac{1}{2}(1+\alpha_2-\zeta_1+2\xi)W_i\,, \\
\overset{4}{g}_{00} &= -2\beta U^2+(2+2\gamma+\alpha_3+\zeta_1-2\xi)\Phi_1+2(1+3\gamma-2\beta+\zeta_2+\xi)\Phi_2 \\
&\quad +2(1+\zeta_3)\Phi_3+2(3\gamma+3\zeta_4-2\xi)\Phi_4-2\xi\Phi_W-(\zeta_1-2\xi)\mathcal{A}\,,
\end{aligned}$$

Standard post-Newtonian gauge

- PPN formalism assumes fixed standard gauge.
- Metric in standard PPN gauge:

$$
\begin{aligned}
\overset{2}{g}_{00} &= 2U\,, \\
\overset{2}{g}_{ij} &= 2\gamma U\delta_{ij}\,, \\
\overset{3}{g}_{0i} &= -\frac{1}{2}(3 + 4\gamma + \alpha_1 - \alpha_2 + \zeta_1 - 2\xi)V_i - \frac{1}{2}(1 + \alpha_2 - \zeta_1 + 2\xi)W_i\,, \\
\overset{4}{g}_{00} &= -2\beta U^2 + (2 + 2\gamma + \alpha_3 + \zeta_1 - 2\xi)\Phi_1 + 2(1 + 3\gamma - 2\beta + \zeta_2 + \xi)\Phi_2 \\
&\quad + 2(1 + \zeta_3)\Phi_3 + 2(3\gamma + 3\zeta_4 - 2\xi)\Phi_4 - 2\xi\Phi_W - (\zeta_1 - 2\xi)\mathcal{A}\,,
\end{aligned}
$$

- Metric contains PPN parameters and PPN potentials.

Standard post-Newtonian gauge

- PPN formalism assumes fixed standard gauge.
- Metric in standard PPN gauge:

$$\begin{aligned}
\overset{2}{g}_{00} &= 2U\,,\\
\overset{2}{g}_{ij} &= 2\gamma U\delta_{ij}\,,\\
\overset{3}{g}_{0i} &= -\frac{1}{2}(3+4\gamma+\alpha_1-\alpha_2+\zeta_1-2\xi)V_i-\frac{1}{2}(1+\alpha_2-\zeta_1+2\xi)W_i\,,\\
\overset{4}{g}_{00} &= -2\beta U^2+(2+2\gamma+\alpha_3+\zeta_1-2\xi)\Phi_1+2(1+3\gamma-2\beta+\zeta_2+\xi)\Phi_2\\
&\quad+2(1+\zeta_3)\Phi_3+2(3\gamma+3\zeta_4-2\xi)\Phi_4-2\xi\Phi_W-(\zeta_1-2\xi)\mathcal{A}\,,
\end{aligned}$$

- Metric contains PPN parameters and PPN potentials.
- Properties of standard PPN metric:
  - Second-order spatial part $\overset{2}{g}_{ij}$ is diagonal.

Standard post-Newtonian gauge

- PPN formalism assumes fixed standard gauge.
- Metric in standard PPN gauge:

$$\begin{aligned}
\overset{2}{g}_{00} &= 2U\,,\\
\overset{2}{g}_{ij} &= 2\gamma U\delta_{ij}\,,\\
\overset{3}{g}_{0i} &= -\frac{1}{2}(3+4\gamma+\alpha_1-\alpha_2+\zeta_1-2\xi)V_i-\frac{1}{2}(1+\alpha_2-\zeta_1+2\xi)W_i\,,\\
\overset{4}{g}_{00} &= -2\beta U^2+(2+2\gamma+\alpha_3+\zeta_1-2\xi)\Phi_1+2(1+3\gamma-2\beta+\zeta_2+\xi)\Phi_2\\
&\quad+2(1+\zeta_3)\Phi_3+2(3\gamma+3\zeta_4-2\xi)\Phi_4-2\xi\Phi_W-(\zeta_1-2\xi)\mathcal{A}\,,
\end{aligned}$$

- Metric contains PPN parameters and PPN potentials.
- Properties of standard PPN metric:
  - Second-order spatial part $\overset{2}{g}_{ij}$ is diagonal.
  - Fourth-order temporal part $\overset{4}{g}_{00}$ does not contain potential $\mathcal{B}$.

PPN parameters

- PPN parameters are linked to physical properties:
    - γ: spatial curvature generated by unit mass.
    - β: non-linearity in gravity superposition law.
    - $\alpha_1, \alpha_2, \alpha_3$: violation of local Lorentz invariance.
    - $\alpha_3, \zeta_1, \zeta_2, \zeta_3, \zeta_4$: violation of energy-momentum conservation.
    - ξ: violation of local position invariance.

PPN parameters

- PPN parameters are linked to physical properties:
  - γ: spatial curvature generated by unit mass.
  - β: non-linearity in gravity superposition law.
  - $\alpha_1, \alpha_2, \alpha_3$: violation of local Lorentz invariance.
  - $\alpha_3, \zeta_1, \zeta_2, \zeta_3, \zeta_4$: violation of energy-momentum conservation.
  - ξ: violation of local position invariance.
- In GR $\gamma = \beta = 1$ and $\xi = \alpha_1 = \alpha_2 = \alpha_3 = \zeta_1 = \zeta_2 = \zeta_3 = \zeta_4 = 0$.

PPN parameters

- PPN parameters are linked to physical properties:
  - γ: spatial curvature generated by unit mass.
  - β: non-linearity in gravity superposition law.
  - $\alpha_1, \alpha_2, \alpha_3$: violation of local Lorentz invariance.
  - $\alpha_3, \zeta_1, \zeta_2, \zeta_3, \zeta_4$: violation of energy-momentum conservation.
  - ξ: violation of local position invariance.
- In GR $\gamma = \beta = 1$ and $\xi = \alpha_1 = \alpha_2 = \alpha_3 = \zeta_1 = \zeta_2 = \zeta_3 = \zeta_4 = 0$.

⇒ Fully conservative gravity theory:
  - No preferred frame or preferred location effects.
  - Total energy-momentum is conserved.

PPN parameters

- PPN parameters are linked to physical properties:
  - γ: spatial curvature generated by unit mass.
  - β: non-linearity in gravity superposition law.
  - $\alpha_1, \alpha_2, \alpha_3$: violation of local Lorentz invariance.
  - $\alpha_3, \zeta_1, \zeta_2, \zeta_3, \zeta_4$: violation of energy-momentum conservation.
  - ξ: violation of local position invariance.
- In GR $\gamma = \beta = 1$ and $\xi = \alpha_1 = \alpha_2 = \alpha_3 = \zeta_1 = \zeta_2 = \zeta_3 = \zeta_4 = 0$.

⇒ Fully conservative gravity theory:

  - No preferred frame or preferred location effects.
  - Total energy-momentum is conserved.

⇒ Other theories will receive bounds from experiments.

Experimental bounds

Par.	Bound	Effects	Experiment
$\gamma - 1$	$2.3 \cdot 10^{-5}$	Time delay, light deflection	Cassini tracking
$\beta - 1$	$8 \cdot 10^{-5}$	Perihelion shift	Perihelion shift
ξ	$4 \cdot 10^{-9}$	Spin precession	Millisecond pulsars
α_1	10^{-4}	Orbital polarization	Lunar laser ranging
α_1	$4 \cdot 10^{-5}$	Orbital polarization	PSR J1738+0333
α_2	$2 \cdot 10^{-9}$	Spin precession	Millisecond pulsars
α_3	$4 \cdot 10^{-20}$	Self-acceleration	Pulsar spin-down statistics
η_N[1]	$9 \cdot 10^{-4}$	Nordtvedt effect	Lunar Laser Ranging
ζ_1	0.02	Combined PPN bounds	—
ζ_2	$4 \cdot 10^{-5}$	Binary pulsar acceleration	PSR 1913+16
ζ_3	10^{-8}	Newton's 3rd law	Lunar acceleration
ζ_4	0.006	—	Kreuzer experiment

---

[1] $\eta_N = 4\beta - \gamma - 3 - \frac{10}{3}\xi - \alpha_1 + \frac{2}{3}\alpha_2 - \frac{2}{3}\zeta_1 - \frac{1}{3}\zeta_2$

PPN potentials

- Newtonian potential:

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

PPN potentials

- Newtonian potential:

$$\chi = -\int \mathrm{d}^3x' \rho' |\vec{x} - \vec{x}'| \,, \quad U = \int \mathrm{d}^3x' \frac{\rho'}{|\vec{x} - \vec{x}'|} \,, \quad \rho' \equiv \rho(t, \vec{x}') \,.$$

- Vector potentials:

$$V_i = \int d^3x' \frac{\rho' v_i'}{|\vec{x} - \vec{x}'|} \,, \quad W_i = \int d^3x' \frac{\rho' v_j'(x_i - x_i')(x_j - x_j')}{|\vec{x} - \vec{x}'|^3} \,.$$

PPN potentials

- Newtonian potential:

$$\chi = -\int \mathrm{d}^3x' \rho' |\vec{x} - \vec{x}'| , \quad U = \int \mathrm{d}^3x' \frac{\rho'}{|\vec{x} - \vec{x}'|} , \quad \rho' \equiv \rho(t, \vec{x}') .$$

- Vector potentials:

$$V_i = \int d^3x' \frac{\rho' v_i'}{|\vec{x} - \vec{x}'|} , \quad W_i = \int d^3x' \frac{\rho' v_j'(x_i - x_i')(x_j - x_j')}{|\vec{x} - \vec{x}'|^3} .$$

- Fourth-order scalar potentials:

$$\Phi_1 = \int d^3x' \frac{\rho' v'^2}{|\vec{x} - \vec{x}'|} , \quad \Phi_4 = \int d^3x' \frac{p'}{|\vec{x} - \vec{x}'|} ,$$

$$\Phi_2 = \int d^3x' \frac{\rho' U'}{|\vec{x} - \vec{x}'|} , \quad \mathcal{A} = \int d^3x' \frac{\rho' \left[v_i'(x_i - x_i')\right]^2}{|\vec{x} - \vec{x}'|^3} ,$$

$$\Phi_3 = \int d^3x' \frac{\rho' \Pi'}{|\vec{x} - \vec{x}'|} , \quad \mathcal{B} = \int d^3x' \frac{\rho'}{|\vec{x} - \vec{x}'|}(x_i - x_i') \frac{dv_i'}{dt} ,$$

$$\Phi_W = \int d^3x' d^3x'' \rho' \rho'' \frac{x_i - x_i'}{|\vec{x} - \vec{x}'|^3} \left(\frac{x_i' - x_i''}{|\vec{x} - \vec{x}''|} - \frac{x_i - x_i''}{|\vec{x}' - \vec{x}''|} \right) .$$

Post-Newtonian field equations

- Expand energy-momentum tensor in velocity orders:

$$\begin{aligned}\Theta_{00} &= \rho\left(1 - \overset{2}{g}_{00} + v^2 + \Pi\right) + \mathcal{O}(6)\,,\\ \Theta_{0i} &= -\rho v_i + \mathcal{O}(5)\,,\\ \Theta_{ij} &= \rho v_i v_j + p\delta_{ij} + \mathcal{O}(6)\end{aligned}$$

Post-Newtonian field equations

- Expand energy-momentum tensor in velocity orders:

$$\begin{aligned}\Theta_{00} &= \rho\left(1 - \overset{2}{g}_{00} + v^2 + \Pi\right) + \mathcal{O}(6)\,,\\ \Theta_{0i} &= -\rho v_i + \mathcal{O}(5)\,,\\ \Theta_{ij} &= \rho v_i v_j + p\delta_{ij} + \mathcal{O}(6)\end{aligned}$$

- Energy-momentum tensor ~ derivatives of PPN potentials.

Post-Newtonian field equations

- Expand energy-momentum tensor in velocity orders:

$$\begin{aligned}\Theta_{00} &= \rho\left(1 - \overset{2}{g}_{00} + v^2 + \Pi\right) + \mathcal{O}(6)\,,\\ \Theta_{0i} &= -\rho v_i + \mathcal{O}(5)\,,\\ \Theta_{ij} &= \rho v_i v_j + p\delta_{ij} + \mathcal{O}(6)\end{aligned}$$

- Energy-momentum tensor ~ derivatives of PPN potentials.

⇒ Solve for PPN parameters by PPN expanding field equations.

Post-Newtonian field equations

- Expand energy-momentum tensor in velocity orders:

$$\begin{aligned}\Theta_{00} &= \rho\left(1 - \overset{2}{g}_{00} + v^2 + \Pi\right) + \mathcal{O}(6)\,,\\ \Theta_{0i} &= -\rho v_i + \mathcal{O}(5)\,,\\ \Theta_{ij} &= \rho v_i v_j + p\delta_{ij} + \mathcal{O}(6)\end{aligned}$$

- Energy-momentum tensor ~ derivatives of PPN potentials.

⇒ Solve for PPN parameters by PPN expanding field equations.

↯ Equations may be coupled to each other, lengthy & hard to solve.

Post-Newtonian field equations

- Expand energy-momentum tensor in velocity orders:

$$\begin{aligned}\Theta_{00} &= \rho\left(1 - \overset{2}{g}_{00} + v^2 + \Pi\right) + \mathcal{O}(6)\,,\\ \Theta_{0i} &= -\rho v_i + \mathcal{O}(5)\,,\\ \Theta_{ij} &= \rho v_i v_j + p\delta_{ij} + \mathcal{O}(6)\end{aligned}$$

- Energy-momentum tensor ~ derivatives of PPN potentials.

⇒ Solve for PPN parameters by PPN expanding field equations.

↯ Equations may be coupled to each other, lengthy & hard to solve.

⇝ Use tensor computer algebra to simplify and solve equations.

Post-Newtonian field equations

- Expand energy-momentum tensor in velocity orders:

$$\begin{aligned}\Theta_{00} &= \rho\left(1 - \overset{2}{g}_{00} + v^2 + \Pi\right) + \mathcal{O}(6)\,,\\ \Theta_{0i} &= -\rho v_i + \mathcal{O}(5)\,,\\ \Theta_{ij} &= \rho v_i v_j + p\delta_{ij} + \mathcal{O}(6)\end{aligned}$$

- Energy-momentum tensor ~ derivatives of PPN potentials.

⇒ Solve for PPN parameters by PPN expanding field equations.

↯ Equations may be coupled to each other, lengthy & hard to solve.

⇝ Use tensor computer algebra to simplify and solve equations.

⇝ Implement generic PPN formalism in *xAct*/*xTensor*: *xPPN*.

Outline

Basic concepts of *xPPN*

- Consider spacetime as product manifold $M_4 = T_1 \times S_3$.

Basic concepts of *xPPN*

- Consider spacetime as product manifold $M_4 = T_1 \times S_3$.
- Proper $3 + 1$ split of tensors on the spacetime manifold:
  - Defining a vector field A_μ on $M_4 \Rightarrow$ scalar field A_0 and vector field A_i on S_3.
  - Split tensor fields depend on S_3 and time parameter.
  - Splitting algorithm takes into account symmetries of tensors defined on M_4.
  - Derivatives on M_4 decompose into derivatives on S_3 and time derivatives.

Basic concepts of *xPPN*

- Consider spacetime as product manifold $M_4 = T_1 \times S_3$.
- Proper $3 + 1$ split of tensors on the spacetime manifold:
  - Defining a vector field A_μ on $M_4 \Rightarrow$ scalar field A_0 and vector field A_i on S_3.
  - Split tensor fields depend on S_3 and time parameter.
  - Splitting algorithm takes into account symmetries of tensors defined on M_4.
  - Derivatives on M_4 decompose into derivatives on S_3 and time derivatives.
- Perturbative expansion of tensors in velocity orders:
  - Perturbative expansion $A_i = \overset{0}{A}_i + \overset{1}{A}_i + \ldots$ automatically defined for every tensor field.
  - Proper counting of perturbation orders in sums, products, scalar functions…
  - Time derivatives are taken into account and weighted with additional velocity order.

Basic concepts of *xPPN*

- Consider spacetime as product manifold $M_4 = T_1 \times S_3$.
- Proper $3 + 1$ split of tensors on the spacetime manifold:
  - Defining a vector field A_μ on $M_4 \Rightarrow$ scalar field A_0 and vector field A_i on S_3.
  - Split tensor fields depend on S_3 and time parameter.
  - Splitting algorithm takes into account symmetries of tensors defined on M_4.
  - Derivatives on M_4 decompose into derivatives on S_3 and time derivatives.
- Perturbative expansion of tensors in velocity orders:
  - Perturbative expansion $A_i = \overset{0}{A}_i + \overset{1}{A}_i + \ldots$ automatically defined for every tensor field.
  - Proper counting of perturbation orders in sums, products, scalar functions. . .
  - Time derivatives are taken into account and weighted with additional velocity order.
- Pre-defined objects ready to use together with standard relations among them:
  - Background manifolds M_4, S_3, T_1 and time parameter.
  - Background geometry: metrics $\eta_{\mu\nu}$ on M_4 and δ_{ij} on S_3.
  - Dynamical geometry: metric $g_{\mu\nu}$, tetrad $\theta^\Gamma{}_\mu$, covariant derivatives $\overset{\circ}{\nabla}$, $\overset{\bullet}{\nabla}$, $\overset{\times}{\nabla}$.
  - PPN potentials: $\chi, U, U_{ij}, V_i, W_i, \Phi_1, \Phi_2, \Phi_3, \Phi_4, \Phi_W, \mathcal{A}, \mathcal{B}$.
  - PPN parameters: $\beta, \gamma, \alpha_1, \alpha_2, \alpha_3, \zeta_1, \zeta_2, \zeta_3, \zeta_4, \xi$.
  - Energy-momentum variables: $\Theta_{\mu\nu}$, ρ, v^i, Π, p.

Manifolds, bundles and indices

- Pre-defined manifolds:
    1. `MfSpacetime`: spacetime manifold M_4.
    2. `MfSpace`: space manifold S_3.
    3. `MfTime`: time manifold T_1.

Manifolds, bundles and indices

- Pre-defined manifolds:
  1. `MfSpacetime`: spacetime manifold M_4.
  2. `MfSpace`: space manifold S_3.
  3. `MfTime`: time manifold T_1.
- Every manifold canonically equipped with tangent bundle:
  1. `TangentMfSpacetime`: tangent bundle of spacetime manifold $\mathbb{T}M_4$.
  2. `TangentMfSpace`: tangent bundle of space manifold $\mathbb{T}S_3$.
  3. `TangentMfTime`: tangent bundle of time manifold $\mathbb{T}T_1$.

Manifolds, bundles and indices

- Pre-defined manifolds:
  1. `MfSpacetime`: spacetime manifold M_4.
  2. `MfSpace`: space manifold S_3.
  3. `MfTime`: time manifold T_1.
- Every manifold canonically equipped with tangent bundle:
  1. `TangentMfSpacetime`: tangent bundle of spacetime manifold $\mathbb{T}M_4$.
  2. `TangentMfSpace`: tangent bundle of space manifold $\mathbb{T}S_3$.
  3. `TangentMfTime`: tangent bundle of time manifold $\mathbb{T}T_1$.
- Lorentz bundle defined for all manifolds (used for tetrads):
  1. `LorentzMfSpacetime`: Lorentz bundle of spacetime manifold $\mathbb{L}M_4$.
  2. `LorentzMfSpace`: Lorentz bundle of space manifold $\mathbb{L}S_3$.
  3. `LorentzMfTime`: Lorentz bundle of time manifold $\mathbb{L}T_1$.

Manifolds, bundles and indices

- Pre-defined manifolds:
  1. `MfSpacetime`: spacetime manifold M_4.
  2. `MfSpace`: space manifold S_3.
  3. `MfTime`: time manifold T_1.
- Every manifold canonically equipped with tangent bundle:
  1. `TangentMfSpacetime`: tangent bundle of spacetime manifold $\mathbb{T}M_4$.
  2. `TangentMfSpace`: tangent bundle of space manifold $\mathbb{T}S_3$.
  3. `TangentMfTime`: tangent bundle of time manifold $\mathbb{T}T_1$.
- Lorentz bundle defined for all manifolds (used for tetrads):
  1. `LorentzMfSpacetime`: Lorentz bundle of spacetime manifold $\mathbb{L}M_4$.
  2. `LorentzMfSpace`: Lorentz bundle of space manifold $\mathbb{L}S_3$.
  3. `LorentzMfTime`: Lorentz bundle of time manifold $\mathbb{L}T_1$.
- Indices defined for all bundles and object types:
  1. `LI[0]`: time component of the $3 + 1$ decomposed forms of tensors.
  2. `\[ScriptT]` (printed as t): index on the tangent bundle $\mathbb{T}T_1$.
  3. `\[ScriptCapitalT]` (printed as $\mathcal{T}$): index on the Lorentz bundle $\mathbb{L}T_1$.
  4. Lowercase Latin letters $a, \ldots, z$, input as `T3a`, . . . , `T3z`: indices on $\mathbb{T}S_3$.
  5. Uppercase Latin letters $A, \ldots, Z$, input as `L3A`, . . . , `L3Z`: indices on $\mathbb{L}S_3$.
  6. Lowercase Greek letters $\alpha, \ldots, \omega$, input as `T4α`, . . . , `T4ω`: indices on $\mathbb{T}M_4$.
  7. Uppercase Greek letters $A, \ldots, \Omega$, input as `L4A`, . . . , `L4Ω`: indices on $\mathbb{L}M_4$.

Background geometry

- Pre-defined objects representing background geometry:

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	$(-\mathbb{L}T_1, \mathbb{T}T_1)$

Background geometry

- Pre-defined objects representing background geometry:

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	$(-\mathbb{L}T_1, \mathbb{T}T_1)$

! Note that background objects are used on indices:

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


In[]:= ToCanonical[SeparateMetric[][A[-T4α] A[T4α]]]
```

Out[]= $A^{\alpha}A^{\beta}\eta_{\alpha\beta}$

Pre-defined dynamical geometry

- Metric tensor and its inverse:
  - Metric $g_{\alpha\beta}$ written as `Met[-T4α, -T4β]`.
  - Inverse metric $g^{\alpha\beta}$ written as `InvMet[T4α, T4β]`.
  - ! Metric and its inverse are different objects, since indices are raised with η:

```
In[]:= ToCanonical[SeparateMetric[][Met[T4α, T4β]]]
```

Out[]= $\eta^{\alpha\gamma}\eta^{\beta\delta}g_{\gamma\delta}$

Pre-defined dynamical geometry

- Metric tensor and its inverse:
  - Metric $g_{\alpha\beta}$ written as `Met[-T4α, -T4β]`.
  - Inverse metric $g^{\alpha\beta}$ written as `InvMet[T4α, T4β]`.
  - ! Metric and its inverse are different objects, since indices are raised with η:

```
In[]:= ToCanonical[SeparateMetric[][Met[T4α, T4β]]]
```

Out[]= $\eta^{\alpha\gamma}\eta^{\beta\delta}g_{\gamma\delta}$

- Levi-Civita covariant derivative $\mathring{\nabla}$ and Christoffel symbols:
  - Covariant derivative $\mathring{\nabla}_{\alpha}$ denoted by `CD[-T4α]`.
  - Christoffel symbols $\mathring{\Gamma}^{\alpha}{}_{\gamma\beta}$ denoted by `ChristoffelCD[T4α, -T4γ, -T4β]`.
  - ! Note order of indices used by *xAct* in conversion:

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

In[]:= CD[-T4γ][A[T4α]]
```

Out[]= $\mathring{\nabla}_{\gamma}A^{\alpha}$

```
In[]:= ChangeCovD[%, CD, PD]
```

Out[]= $\partial_{\gamma}A^{\alpha}+\mathring{\Gamma}^{\alpha}{}_{\gamma\beta}A^{\beta}$

$3 + 1$ split and perturbative expansion of tensor fields

- Function `PPN` to extract $3 + 1$ decomposition and velocity orders:
  - `PPN[`*h*`][`*i*`]` extracts $3 + 1$ split of tensor *h* with indices *i*:

```
In[]:= PPN[Met][-LI[0], -T3a]
```

Out[]= g_{0a}

  - `PPN[`*h*`, `*n*`][`*i*`]` extracts *n*'th order perturbation of tensor *h* with indices *i*:

```
In[]:= PPN[Met, 3][-LI[0], -T3a]
```

Out[]= $\overset{3}{g}_{0a}$

$3+1$ split and perturbative expansion of tensor fields

- Function `PPN` to extract $3+1$ decomposition and velocity orders:
  - `PPN[`*h*`][`*i*`]` extracts $3+1$ split of tensor *h* with indices *i*:

```
In[]:= PPN[Met][-LI[0], -T3a]
```

Out[]= g_{0a}

  - `PPN[`*h*`, `*n*`][`*i*`]` extracts *n*'th order perturbation of tensor *h* with indices *i*:

```
In[]:= PPN[Met, 3][-LI[0], -T3a]
```

Out[]= $\overset{3}{g}_{0a}$

- Arguments to `PPN`:
  - *h* can be any (pre-defined or custom) tensor head.
  - *n* is a non-negative integer perturbation order.
  - *i* is a sequence of indices either on S_3 or `LI[0]` (time component).

Using symmetries of tensor fields

- Example: define an antisymmetric tensor field $A_{\alpha\beta} = A_{[\alpha\beta]}$:

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

Using symmetries of tensor fields

- Example: define an antisymmetric tensor field $A_{\alpha\beta} = A_{[\alpha\beta]}$:

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

- Automatically defined $3+1$ split respects symmetries of original tensor:
  - Vanishing component $A_{00} = 0$ evaluates to zero:

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

  - Independent component A_{0a} remains unevaluated:

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

  - Dependent component $A_{a0} = -A_{0a}$ automatically evaluates to independent component:

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

  - Indices on antisymmetric components $A_{ba} = -A_{ab}$ can be ordered with `ToCanonical`:

```
In[]:= ToCanonical[PPN[A][-T3b, -T3a]]
Out[]= $-A_{ab}$
```

3 + 1 split of arbitrary expressions

- Example: consider tensor field $A^{\alpha}{}_{\beta}$ with mixed indices:

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

$3 + 1$ split of arbitrary expressions

- Example: consider tensor field $A^{\alpha}{}_{\beta}$ with mixed indices:

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

- Using `SpaceTimeSplit` on single tensor yields tensor components:

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

Out[]= $A^{a}{}_{0}$

$3 + 1$ split of arbitrary expressions

- Example: consider tensor field $A^{\alpha}{}_{\beta}$ with mixed indices:

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

- Using `SpaceTimeSplit` on single tensor yields tensor components:

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

Out[]= $A^{a}{}_{0}$

- On compound expressions, also dummy indices are split:

```
In[]:= SpaceTimeSplit[A[T4α, -T4γ] A[T4γ, -T4β],
    {T4α → T3a, -T4β → -LI[0]}]
```

Out[]= $A^{a}{}_{0}A^{0}{}_{0} + A^{a}{}_{b}A^{b}{}_{0}$

$3 + 1$ split of arbitrary expressions

- Example: consider tensor field $A^{\alpha}{}_{\beta}$ with mixed indices:

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

- Using `SpaceTimeSplit` on single tensor yields tensor components:

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

Out[]= $A^{a}{}_{0}$

- On compound expressions, also dummy indices are split:

```
In[]:= SpaceTimeSplit[A[T4α, -T4γ] A[T4γ, -T4β],
    {T4α → T3a, -T4β → -LI[0]}]
```

Out[]= $A^{a}{}_{0}A^{0}{}_{0} + A^{a}{}_{b}A^{b}{}_{0}$

- Use `SpaceTimeSplits` to obtain all combinations of space and time:

```
In[]:= SpaceTimeSplits[A[T4α, -T4β], {T4α → T3a, -T4β → -T3b}]
```

Out[]= $\{\{A^{0}{}_{0},\ A^{0}{}_{b}\},\ \{A^{a}{}_{0},\ A^{a}{}_{b}\}\}$

Perturbative expansion of arbitrary expressions

- Example: consider tensor field ${A^{\alpha}}_{\beta}$ with mixed indices:

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

Perturbative expansion of arbitrary expressions

- Example: consider tensor field $A^{\alpha}{}_{\beta}$ with mixed indices:

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

- Use `VelocityOrder` on single tensor component to get perturbation:

```
In[]:= VelocityOrder[PPN[A][T3a, -T3b], 3]
```

Out[]= $\overset{3}{A}{}^{a}{}_{b}$

Perturbative expansion of arbitrary expressions

- Example: consider tensor field ${A^\alpha}_\beta$ with mixed indices:

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

- Use `VelocityOrder` on single tensor component to get perturbation:

```
In[]:= VelocityOrder[PPN[A][T3a, -T3b], 3]
```

Out[]= ${\overset{3}{A}{}^a}_b$

- On compound expressions, the product rule is obeyed:

```
In[]:= VelocityOrder[PPN[A][T3a, -T3c] PPN[A][T3c, -T3b], 2]
```

Out[]= ${\overset{0}{A}{}^a}_c {\overset{2}{A}{}^c}_b + {\overset{1}{A}{}^a}_c {\overset{1}{A}{}^c}_b + {\overset{2}{A}{}^a}_c {\overset{0}{A}{}^c}_b$

Perturbative expansion of arbitrary expressions

- Example: consider tensor field $A^{\alpha}{}_{\beta}$ with mixed indices:

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

- Use `VelocityOrder` on single tensor component to get perturbation:

```
In[]:= VelocityOrder[PPN[A][T3a, -T3b], 3]
```

Out[]= $\overset{3}{A}{}^{a}{}_{b}$

- On compound expressions, the product rule is obeyed:

```
In[]:= VelocityOrder[PPN[A][T3a, -T3c] PPN[A][T3c, -T3b], 2]
```

Out[]= $\overset{0}{A}{}^{a}{}_{c}\overset{2}{A}{}^{c}{}_{b} + \overset{1}{A}{}^{a}{}_{c}\overset{1}{A}{}^{c}{}_{b} + \overset{2}{A}{}^{a}{}_{c}\overset{0}{A}{}^{c}{}_{b}$

- Partial spatial derivatives are transparent to perturbative expansion:

```
In[]:= VelocityOrder[PD[-T3c][PPN[A][T3a, -T3b]], 1]
```

Out[]= $\partial_{c}\overset{1}{A}{}^{a}{}_{b}$

Pre-defined rules for metric perturbative expansion

- Zeroth-order metric perturbations automatically evaluate to background:

```
In[]:= SpaceTimeSplits[Met[-T4α, -T4β],
    {-T4α → -T3a, -T4β → -T3b}]
Out[]= {{g00, g0b}, {g0a, gab}}

In[]:= Map[VelocityOrder[#, 0] &, %, {2}]
Out[]= {{-1, 0}, {0, δab}}
```

Pre-defined rules for metric perturbative expansion

- Zeroth-order metric perturbations automatically evaluate to background:

```
In[]:= SpaceTimeSplits[Met[-T4α, -T4β],
    {-T4α → -T3a, -T4β → -T3b}]
Out[]= {{g00, g0b}, {g0a, gab}}

In[]:= Map[VelocityOrder[#, 0] &, %, {2}]
Out[]= {{-1, 0}, {0, δab}}
```

- All other metric perturbations vanish except:

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

Pre-defined rules for metric perturbative expansion

- Zeroth-order metric perturbations automatically evaluate to background:

```
In[]:= SpaceTimeSplits[Met[-T4α, -T4β],
    {-T4α → -T3a, -T4β → -T3b}]
```

Out[]= $\{\{g_{00},\ g_{0b}\},\ \{g_{0a},\ g_{ab}\}\}$

```
In[]:= Map[VelocityOrder[#, 0] &, %, {2}]
```

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

- All other metric perturbations vanish except:

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

- Inverse metric is expanded automatically:

```
In[]:= VelocityOrder[PPN[InvMet][T3a, T3b], 4]
```

Out[]= $\overset{2}{g}{}^{ac}\overset{2}{g}{}^{b}{}_{c} - \overset{4}{g}{}^{ab}$

Further pre-defined rules for perturbative expansion

- Christoffel symbols expand to derivatives of the metric:

```
In[]:= VelocityOrder[PPN[ChristoffelCD][T3a, -T3c, -T3b], 2]
```

Out[]= $\frac{1}{2}\partial_b\overset{2}{g}_c{}^a + \frac{1}{2}\partial_c\overset{2}{g}_b{}^a - \frac{1}{2}\partial^a\overset{2}{g}_{cb}$

Further pre-defined rules for perturbative expansion

- Christoffel symbols expand to derivatives of the metric:

```
In[]:= VelocityOrder[PPN[ChristoffelCD][T3a, -T3c, -T3b], 2]
```

Out[]= $\frac{1}{2}\partial_b \overset{2}{g}_c{}^a + \frac{1}{2}\partial_c \overset{2}{g}_b{}^a - \frac{1}{2}\partial^a \overset{2}{g}_{cb}$

- Curvature tensors likewise expand into metric derivatives:

```
In[]:= VelocityOrder[PPN[RicciCD][-LI[0], -LI[0]], 2]
```

Out[]= $-\frac{1}{2}\partial_a \partial^a \overset{2}{g}_{00}$

Further pre-defined rules for perturbative expansion

- Christoffel symbols expand to derivatives of the metric:

```
In[]:= VelocityOrder[PPN[ChristoffelCD][T3a, -T3c, -T3b], 2]
```

Out[]= $\frac{1}{2}\partial_b\overset{2}{g}_c{}^a + \frac{1}{2}\partial_c\overset{2}{g}_b{}^a - \frac{1}{2}\partial^a\overset{2}{g}_{cb}$

- Curvature tensors likewise expand into metric derivatives:

```
In[]:= VelocityOrder[PPN[RicciCD][-LI[0], -LI[0]], 2]
```

Out[]= $-\frac{1}{2}\partial_a\partial^a\overset{2}{g}_{00}$

- Perturbative expansion can also be performed using `ApplyPPNRules`:

```
In[]:= PPN[RicciCD, 2][-LI[0], -LI[0]]
```

Out[]= $\overset{2}{R}_{00}$

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

Out[]= $-\frac{1}{2}\partial_a\partial^a\overset{2}{g}_{00}$

Defining additional rules for perturbative expansion

- Newly defined objects have only generic perturbative expansion:

```
In[]:= DefTensor[A[T4α], {MfSpacetime}]
In[]:= VelocityOrder[PPN[A][LI[0]], 0]
```

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

Defining additional rules for perturbative expansion

- Newly defined objects have only generic perturbative expansion:

```
In[]:= DefTensor[A[T4α], {MfSpacetime}]
In[]:= VelocityOrder[PPN[A][LI[0]], 0]
```

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

- Define a rule with `OrderSet`, which will then be used:

```
In[]:= OrderSet[PPN[A, 0][LI[0]], -1];
In[]:= VelocityOrder[PPN[A][LI[0]], 0]
```

Out[]= -1

Defining additional rules for perturbative expansion

- Newly defined objects have only generic perturbative expansion:

```
In[]:= DefTensor[A[T4α], {MfSpacetime}]
In[]:= VelocityOrder[PPN[A][LI[0]], 0]
```

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

- Define a rule with `OrderSet`, which will then be used:

```
In[]:= OrderSet[PPN[A, 0][LI[0]], -1];
In[]:= VelocityOrder[PPN[A][LI[0]], 0]
```

Out[]= -1

- Automatic expansion of perturbations switched off with `UsePPNRules →False`:

```
In[]:= VelocityOrder[PPN[A][LI[0]], 0, UsePPNRules → False]
```

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

```
In[]:= VelocityOrder[PPN[Met][-LI[0], -LI[0]], 0,
    UsePPNRules → False]
```

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

Treatment of time derivatives

- Pre-defined parameter `TimePar`, printed as 0, to represent time.
- $3 + 1$ split tensor components carry dependence on `MfSpace` and `TimePar`.

Treatment of time derivatives

- Pre-defined parameter `TimePar`, printed as 0, to represent time.
- $3 + 1$ split tensor components carry dependence on `MfSpace` and `TimePar`.
- Time derivatives are converted automatically in $3 + 1$ split:

```
In[]:= SpaceTimeSplit[PD[-T4γ][Met[-T4α, -T4β]],
    {-T4α → -T3a, -T4β → -T3b, -T4γ → -LI[0]}]
Out[]= ∂0gab
In[]:= % == ParamD[TimePar][PPN[Met][-T3a, -T3b]]
Out[]= True
```

Treatment of time derivatives

- Pre-defined parameter `TimePar`, printed as 0, to represent time.
- $3 + 1$ split tensor components carry dependence on `MfSpace` and `TimePar`.
- Time derivatives are converted automatically in $3 + 1$ split:

```
In[]:= SpaceTimeSplit[PD[-T4γ][Met[-T4α, -T4β]],
    {-T4α → -T3a, -T4β → -T3b, -T4γ → -LI[0]}]
```

Out[]= $\partial_0 g_{ab}$

```
In[]:= % == ParamD[TimePar][PPN[Met][-T3a, -T3b]]
Out[]= True
```

- Time derivatives carry additional velocity order:

```
In[]:= VelocityOrder[ParamD[TimePar][PPN[Met][-T3a, -T3b]], 3]
```

Out[]= $\partial_0 \overset{2}{g}_{ab}$

Pre-defined energy-momentum variables

- Energy-momentum tensor and its trace-reversed form are defined:

```
In[]:= EnergyMomentum[-T4α, -T4β]
```

Out[]= $\Theta_{\alpha\beta}$

```
In[]:= TREnergyMomentum[-T4α, -T4β]
```

Out[]= $\Theta_{\alpha\beta} - \frac{1}{2} g_{\alpha\beta} g^{\gamma\delta} \Theta_{\gamma\delta}$

Pre-defined energy-momentum variables

- Energy-momentum tensor and its trace-reversed form are defined:

```
In[]:= EnergyMomentum[-T4α, -T4β]
```

Out[]= $\Theta_{\alpha\beta}$

```
In[]:= TREnergyMomentum[-T4α, -T4β]
```

Out[]= $\Theta_{\alpha\beta} - \frac{1}{2} g_{\alpha\beta} g^{\gamma\delta} \Theta_{\gamma\delta}$

- Furher energy-momentum variables are also pre-defined:
  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 $v^a \sim \mathcal{O}(1)$.

Pre-defined energy-momentum variables

- Energy-momentum tensor and its trace-reversed form are defined:

`In[]:= EnergyMomentum[-T4α, -T4β]`

Out[]= $\Theta_{\alpha\beta}$

`In[]:= TREnergyMomentum[-T4α, -T4β]`

Out[]= $\Theta_{\alpha\beta} - \frac{1}{2} g_{\alpha\beta} g^{\gamma\delta} \Theta_{\gamma\delta}$

- Furher energy-momentum variables are also pre-defined:
  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 $v^a \sim \mathcal{O}(1)$.
- Perturbative expansion of energy-momentum is performed automatically:

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

Euler equations

- Energy-momentum tensor satisfies covariant conservation equation $\overset{\circ}{\nabla}_{\mu}\Theta^{\mu\nu} = 0$.

Euler equations

- Energy-momentum tensor satisfies covariant conservation equation $\overset{\circ}{\nabla}_{\mu}\Theta^{\mu\nu} = 0$.

⇒ Matter variables satisfy Euler equations, which are applied as follows:

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

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

2. `TimeVelToEuler[`X`]` applies the replacement

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

3. `TimePiToEuler[`X`]` applies the replacement

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

PPN potentials and parameters

Symbol	Pot.
`PotentialChi[]`	χ
`PotentialU[]`	U
`PotentialUU[-T3a, -T3b]`	U_{ab}
`PotentialV[-T3a]`	V_a
`PotentialW[-T3a]`	W_a
`PotentialPhi1[]`	Φ_1
`PotentialPhi2[]`	Φ_2
`PotentialPhi3[]`	Φ_3
`PotentialPhi4[]`	Φ_4
`PotentialPhiW[]`	Φ_W
`PotentialA[]`	$\mathcal{A}$
`PotentialB[]`	$\mathcal{B}$

PPN potentials and parameters

Symbol	Pot.
`PotentialChi[]`	χ
`PotentialU[]`	U
`PotentialUU[-T3a, -T3b]`	U_{ab}
`PotentialV[-T3a]`	V_a
`PotentialW[-T3a]`	W_a
`PotentialPhi1[]`	Φ_1
`PotentialPhi2[]`	Φ_2
`PotentialPhi3[]`	Φ_3
`PotentialPhi4[]`	Φ_4
`PotentialPhiW[]`	Φ_W
`PotentialA[]`	$\mathcal{A}$
`PotentialB[]`	$\mathcal{B}$

Symbol	Par.
`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

Transformation between different 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`	$U_{,00} \to -\frac{1}{2}\triangle(\mathcal{A} + \mathcal{B} - \Phi_1)$

Transformation of PPN potentials to matter variables

Function `PotentialToSource` applies:

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

Sorting of derivatives

- Different order of derivatives required to recognize terms:
    - Matching time derivative requires order $\partial_a\partial_0 A^a$.
    - Matching divergence requires order $\partial_0\partial_a A^a$.

Sorting of derivatives

- Different order of derivatives required to recognize terms:
  - Matching time derivative requires order $\partial_a\partial_0 A^a$.
  - Matching divergence requires order $\partial_0\partial_a A^a$.
- Various functions applied before pattern matching:
  - Sort derivatives in canonical order:

```
In[]:= SortPDs[expr]
```

Out[]= $\partial_0\partial_c\partial_b\partial^b\partial_a A^c$

  - Sort derivatives to time derivatives of tensor A:

```
In[]:= SortPDsToTime[expr, A]
```

Out[]= $\partial^b\partial_a\partial_c\partial_b\partial_0 A^c$

  - Sort derivatives to divergence of tensor A:

```
In[]:= SortPDsToDiv[expr, A]
```

Out[]= $\partial^b\partial_0\partial_a\partial_b\partial_c A^c$

  - Sort derivatives to Laplace operator of tensor A:

```
In[]:= SortPDsToBox[expr, A]
```

Out[]= $\partial_0\partial_a\partial_c\partial_b\partial^b A^c$

Additional variables for teleparallel theories

- Tetrad `Tet` and inverse `InvTet` for teleparallel gravity:
    - Work in Weitzenböck gauge $\omega \equiv 0$ at all perturbation orders.
    - Tetrad perturbations decomposed into symmetric part (metric) and antisymmetric tensor.
    - Additional rules for perturbation of antisymmetric components.

Additional variables for teleparallel theories

- Tetrad `Tet` and inverse `InvTet` for teleparallel gravity:
  - Work in Weitzenböck gauge $\omega \equiv 0$ at all perturbation orders.
  - Tetrad perturbations decomposed into symmetric part (metric) and antisymmetric tensor.
  - Additional rules for perturbation of antisymmetric components.
- Teleparallel (flat, metric) connection $\overset{\bullet}{\nabla}$ implemented as `FD`:
  - Connection coefficients converted into derivatives of the tetrad.
  - Perturbative expansion of torsion and contortion tensors.

Additional variables for teleparallel theories

- Tetrad `Tet` and inverse `InvTet` for teleparallel gravity:
  - Work in Weitzenböck gauge $\omega \equiv 0$ at all perturbation orders.
  - Tetrad perturbations decomposed into symmetric part (metric) and antisymmetric tensor.
  - Additional rules for perturbation of antisymmetric components.
- Teleparallel (flat, metric) connection $\overset{\bullet}{\nabla}$ implemented as `FD`:
  - Connection coefficients converted into derivatives of the tetrad.
  - Perturbative expansion of torsion and contortion tensors.
- Symmetric teleparallel (flat, symmetric) connection $\overset{\times}{\nabla}$ implemented as `ND`:
  - Perturbative expansion around background with vanishing connection coefficients.
  - Perturbations generated by infinitesimal diffeomorphism / vector fields.
  - Perturbative expansion of nonmetricity and disformation tensor fields.

Outline

Action and field equations

- Action of scalar-tensor gravity with massless scalar field: [Nordtvedt '70]

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

Action and field equations

- Action of scalar-tensor gravity with massless scalar field: [Nordtvedt ’70]

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

- Free function $\omega(\psi)$ of the scalar field ψ.

Action and field equations

- Action of scalar-tensor gravity with massless scalar field: [Nordtvedt '70]

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

- Free function $\omega(\psi)$ of the scalar field ψ.
- Jordan frame: no direct coupling between matter and scalar field.

Action and field equations

- Action of scalar-tensor gravity with massless scalar field: [Nordtvedt '70]

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

- Free function $\omega(\psi)$ of the scalar field ψ.
- Jordan frame: no direct coupling between matter and scalar field.

⇒ Field equations:

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

$$(2\omega+3)\mathring{\square}\psi + \frac{d\omega}{d\psi}\partial_\rho\psi\partial^\rho\psi = \kappa^2\Theta\,.$$

Action and field equations

- Action of scalar-tensor gravity with massless scalar field: [Nordtvedt '70]

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

- Free function $\omega(\psi)$ of the scalar field ψ.
- Jordan frame: no direct coupling between matter and scalar field.

⇒ Field equations:

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

$$(2\omega+3)\mathring{\square}\psi + \frac{d\omega}{d\psi}\partial_\rho\psi\partial^\rho\psi = \kappa^2\Theta\,.$$

- Relevant components of scalar field: $\overset{0}{\psi} = \Psi, \overset{2}{\psi}, \overset{4}{\psi}$.

Action and field equations

- Action of scalar-tensor gravity with massless scalar field: [Nordtvedt '70]

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

- Free function $\omega(\psi)$ of the scalar field ψ.
- Jordan frame: no direct coupling between matter and scalar field.

⇒ Field equations:

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

$$(2\omega+3)\overset{\circ}{\square}\psi + \frac{d\omega}{d\psi}\partial_\rho\psi\partial^\rho\psi = \kappa^2\Theta\,.$$

- Relevant components of scalar field: $\overset{0}{\psi} = \Psi, \overset{2}{\psi}, \overset{4}{\psi}$.
- Cosmological background value Ψ assumed to be constant.

Getting started: load the package

1. To start, we must load the *xPPN* package:

```
In[]:= << xAct`xPPN`
```

Getting started: load the package

1. To start, we must load the *xPPN* package:

```
In[]:= << xAct`xPPN`
```

2. Suppress $ symbols in the index notation:

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

Getting started: load the package

1. To start, we must load the *xPPN* package:

```
In[]:= << xAct`xPPN`
```

2. Suppress $ symbols in the index notation:

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

3. Define utility functions to create rules from equations:

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

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

Define geometric objects

1. Scalar field ψ:

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

Define geometric objects

1. Scalar field ψ:

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

2. Cosmological background value Ψ of the scalar field:

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

Define geometric objects

1. Scalar field ψ:

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

2. Cosmological background value Ψ of the scalar field:

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

3. Gravitational constant κ:

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

Define geometric objects

1. Scalar field ψ:

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

2. Cosmological background value Ψ of the scalar field:

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

3. Gravitational constant κ:

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

4. Free function ω of the scalar field:

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

Define placeholders for later use

1. Metric field equations $\mathcal{E}_{\alpha\beta} = 0$:

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

Define placeholders for later use

1. Metric field equations $\mathcal{E}_{\alpha\beta} = 0$:

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

2. Scalar field equations $\mathcal{E} = 0$:

```
In[]:= DefTensor[ScalEq[], {MfSpacetime}, PrintAs → "ℰ"]
```

Define placeholders for later use

1. Metric field equations $\mathcal{E}_{\alpha\beta} = 0$:

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

2. Scalar field equations $\mathcal{E} = 0$:

```
In[]:= DefTensor[ScalEq[], {MfSpacetime}, PrintAs → "ℰ"]
```

3. Constant coefficients to use for solving field equations:

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

Metric field equations

Define metric field equation and save for later use:

```
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] - EnergyMomentum[-T4γ, -T4δ] *
    InvMet[T4γ, T4δ] * Met[-T4α, -T4β] * (omega[psi[]] + 1) /
    (2 omega[psi[]] + 3)) * kappa^2;

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

Scalar field equation

Define scalar field equation and save for later use:

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

Post-Newtonian expansion of the scalar field

1. Define background value $\overset{0}{\psi} = \Psi$:

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

Post-Newtonian expansion of the scalar field

1. Define background value $\overset{0}{\psi} = \Psi$:

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

2. Set first order odd part $\overset{1}{\psi} = 0$:

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

Post-Newtonian expansion of the scalar field

1. Define background value $\overset{0}{\psi} = \Psi$:

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

2. Set first order odd part $\overset{1}{\psi} = 0$:

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

3. Set third order odd part $\overset{3}{\psi} = 0$:

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

3 + 1 split of metric field equations

Use `SpaceTimeSplits` to obtain all components of 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 /@ %];
```

$3+1$ split of scalar field equation

Use `SpaceTimeSplit` to decompose field equation:

```
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[%]];
```

Velocity order decomposition of metric field equations

Use `VelocityOrder` on metric field equations:

```
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 /@ %];
```

Velocity order decomposition of scalar field equation

Use `VelocityOrder` on 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 /@ %];
```

Check zeroth-order (vacuum) field equations

1. Metric field equation (time components) $\overset{0}{\mathcal{E}}_{00} = 0$:

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

Check zeroth-order (vacuum) field equations

1. Metric field equation (time components) $\overset{0}{\mathcal{E}}_{00} = 0$:

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

2. Metric field equation (space components) $\overset{0}{\mathcal{E}}_{ab} = 0$:

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

Check zeroth-order (vacuum) field equations

1. Metric field equation (time components) $\overset{0}{\mathcal{E}}_{00} = 0$:

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

2. Metric field equation (space components) $\overset{0}{\mathcal{E}}_{ab} = 0$:

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

3. Scalar field equation $\overset{0}{\mathcal{E}} = 0$:

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

Extract second-order field equations

1. Extract second-order field equations:

```
In[]:= eqns2 = FullSimplify[{
          PPN[MetEq, 2][-LI[0], -LI[0]],
          PPN[MetEq, 2][-T3a, -T3b],
          PPN[ScalEq, 2][]
      } /. meteqvru /. scaleqvru];
```

Extract second-order field equations

1. Extract second-order field equations:

```
In[]:= eqns2 = FullSimplify[{
        PPN[MetEq, 2][-LI[0], -LI[0]],
        PPN[MetEq, 2][-T3a, -T3b],
        PPN[ScalEq, 2][]
   } /. meteqvru /. scaleqvru];
```

2. Equations take the form:

$$\overset{2}{\mathcal{E}} = \kappa^2\rho + (2\omega(\Psi)+3)\,\triangle\,\overset{2}{\psi}\,,$$

$$\overset{2}{\mathcal{E}}_{00} = -\kappa^2\rho\frac{\omega(\Psi)+2}{2\omega(\Psi)+3} - \frac{\Psi}{2}\,\triangle\,\overset{2}{g}_{00}\,,$$

$$\overset{2}{\mathcal{E}}_{ab} = -\kappa^2\rho\frac{\omega(\Psi)+1}{2\omega(\Psi)+3}\delta_{ab} + \frac{\Psi}{2}\left(\overset{2}{g}_{00,ab} - \overset{2}{g}_{cc,ab} + 2\overset{2}{g}_{c(a,b)c} - \triangle\overset{2}{g}_{ab}\right) - \overset{2}{\psi}_{,ab}\,.$$

Ansatz for second-order field variables

```
In[]:= ans2def = {
         PPN[Met, 2][-LI[0], -LI[0]] == aa[1] * PotentialU[],
         PPN[Met, 2][-T3a, -T3b] == aa[3] * PotentialUU[-T3a, -T3b] +
            aa[2] * PotentialU[] * BkgMetricS3[-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];
```

Use ansatz in second-order field equations

```
In[]:= eqns2 /. ans2ru;
In[]:= PotentialUToChi /@ %;
In[]:= PotentialUUToChi /@ %;
In[]:= Expand[%];
In[]:= ToCanonical /@ %;
In[]:= ContractMetric[#, OverDerivatives → True,
   AllowUpperDerivatives → True] & /@ %;
In[]:= PotentialToSource /@ %;
In[]:= Expand[%];
In[]:= ToCanonical /@ %;
In[]:= SortPDs /@ %;
In[]:= eqnsa2 = FullSimplify[%];
```

Derive and solve equations for constant coefficients a_i

1. Use gauge condition $a_3 = 0$ to obtain unique solution.

Derive and solve equations for constant coefficients a_i

1. Use gauge condition $a_3 = 0$ to obtain unique solution.
2. Extract equations for constant coefficients:

```
In[]:= eqnsc2 = FullSimplify[{
    Coefficient[eqnsa2[[1]], Density[]],
    Coefficient[eqnsa2[[3]], Density[]],
    Coefficient[eqnsa2[[2]], Density[] *
        BkgMetricS3[-T3a, -T3b]],
    aa[3]}];
```

Derive and solve equations for constant coefficients a_i

1. Use gauge condition $a_3 = 0$ to obtain unique solution.
2. Extract equations for constant coefficients:

```
In[]:= eqnsc2 = FullSimplify[{
    Coefficient[eqnsa2[[1]], Density[]],
    Coefficient[eqnsa2[[3]], Density[]],
    Coefficient[eqnsa2[[2]], Density[] *
        BkgMetricS3[-T3a, -T3b]],
    aa[3]}];
```

3. Solve the equations:

```
In[]:= sola2 = FullSimplify[First[Solve[# == 0 & /@ eqnsc2,
    aa /@ Range[1, 4]]]];
```

Derive and solve equations for constant coefficients a_i

1. Use gauge condition $a_3 = 0$ to obtain unique solution.
2. Extract equations for constant coefficients:

```
In[]:= eqnsc2 = FullSimplify[{
    Coefficient[eqnsa2[[1]], Density[]],
    Coefficient[eqnsa2[[3]], Density[]],
    Coefficient[eqnsa2[[2]], Density[] *
        BkgMetricS3[-T3a, -T3b]],
    aa[3]}];
```

3. Solve the equations:

```
In[]:= sola2 = FullSimplify[First[Solve[# == 0 & /@ eqnsc2,
    aa /@ Range[1, 4]]]];
```

4. Solution of the component equations:

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

Check second-order solution

1. Check equations obtained by making ansatz:

```
In[]:= Simplify[eqnsa2 /. sola2]
Out[]= {0, 0, 0}
```

Check second-order solution

1. Check equations obtained by making ansatz:

```
In[]:= Simplify[eqnsa2 /. sola2]
Out[]= {0, 0, 0}
```

2. Insert solution into the perturbations $\overset{2}{g}_{00}, \overset{2}{g}_{ab}, \overset{2}{\psi}$:

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

Check second-order solution

1. Check equations obtained by making ansatz:

```
In[]:= Simplify[eqnsa2 /. sola2]
Out[]= {0, 0, 0}
```

2. Insert solution into the perturbations $\overset{2}{g}_{00}, \overset{2}{g}_{ab}, \overset{2}{\psi}$:

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

3. Check that this result solves the second-order field equations:

```
In[]:= eqns2 /. sol2ru;
In[]:= Expand[%];
In[]:= PotentialToSource /@ %;
In[]:= ToCanonical /@ %;
In[]:= SortPDs /@ %;
In[]:= Simplify[%]
Out[]= {0, 0, 0}
```

Equations at the third velocity order

1. Extract third-order field equations:

```
In[]:= eqns3 = FullSimplify[PPN[MetEq, 3][-LI[0], -T3a]
    /. meteqvru];
```

Equations at the third velocity order

1. Extract third-order field equations:

```
In[]:= eqns3 = FullSimplify[PPN[MetEq, 3][-LI[0], -T3a]
    /. meteqvru];
```

2. This equation takes the form

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

Equations at the third velocity order

1. Extract third-order field equations:

```
In[]:= eqns3 = FullSimplify[PPN[MetEq, 3][-LI[0], -T3a]
    /. meteqvru];
```

2. This equation takes the form

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

3. Define ansatz for the third-order metric perturbation $\overset{3}{g}_{0a}$:

```
In[]:= ans3def = PPN[Met, 3][-LI[0], -T3a] ==
    aa[5] * PotentialV[-T3a] + aa[6] * PotentialW[-T3a]
```

Out[]= $\overset{3}{g}_{0a} = a_5 V_a + a_6 W_a$

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

Insert ansatz into third-order field equations

1. Insert ansatz into field equations:

```
In[]:= eqns3 /. ans3ru /. sol2ru;
In[]:= PotentialWToChiV[%];
In[]:= Expand[%];
In[]:= ContractMetric[%, OverDerivatives → True,
    AllowUpperDerivatives → True];
In[]:= PotentialChiToU[%];
In[]:= PotentialVToU[%];
In[]:= PotentialToSource[%];
In[]:= ToCanonical[%];
In[]:= SortPDs[%];
In[]:= eqnsa3 = FullSimplify[%];
```

Insert ansatz into third-order field equations

1. Insert ansatz into field equations:

```
In[]:= eqns3 /. ans3ru /. sol2ru;
In[]:= PotentialWToChiV[%];
In[]:= Expand[%];
In[]:= ContractMetric[%, OverDerivatives → True,
    AllowUpperDerivatives → True];
In[]:= PotentialChiToU[%];
In[]:= PotentialVToU[%];
In[]:= PotentialToSource[%];
In[]:= ToCanonical[%];
In[]:= SortPDs[%];
In[]:= eqnsa3 = FullSimplify[%];
```

2. Inspecting this equation shows the following form:

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

Solve third-order equations

1. Need gauge condition $a_5 - a_6 = a_0$ with a_0 determined later.

Solve third-order equations

1. Need gauge condition $a_5 - a_6 = a_0$ with a_0 determined later.
2. Solve equations for constant parameters in the ansatz:

```
In[]:= sola3 = FullSimplify[First[Solve[{eqnsa3 == 0,
    aa[6] - aa[5] == aa[0]}, {aa[5], aa[6]}]]];
```

Solve third-order equations

1. Need gauge condition $a_5 - a_6 = a_0$ with a_0 determined later.
2. Solve equations for constant parameters in the ansatz:

```
In[]:= sola3 = FullSimplify[First[Solve[{eqnsa3 == 0,
    aa[6] - aa[5] == aa[0]}, {aa[5], aa[6]}]]];
```

3. The solution is given by

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

Solve third-order equations

1. Need gauge condition $a_5 - a_6 = a_0$ with a_0 determined later.
2. Solve equations for constant parameters in the ansatz:

```
In[]:= sola3 = FullSimplify[First[Solve[{eqnsa3 == 0,
    aa[6] - aa[5] == aa[0]}, {aa[5], aa[6]}]]];
```

3. The solution is given by

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

4. We check that this solves the equations:

```
In[]:= Simplify[eqnsa3 /. sola3]
Out[]= 0
```

Solve third-order equations

1. Need gauge condition $a_5 - a_6 = a_0$ with a_0 determined later.
2. Solve equations for constant parameters in the ansatz:

```
In[]:= sola3 = FullSimplify[First[Solve[{eqnsa3 == 0,
    aa[6] - aa[5] == aa[0]}, {aa[5], aa[6]}]]];
```

3. The solution is given by

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

4. We check that this solves the equations:

```
In[]:= Simplify[eqnsa3 /. sola3]
Out[]= 0
```

5. Insert solution into the ansatz and save for later use:

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

Check third-order field equations

```
In[]:= eqns3 /. 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
```

Equations at the fourth velocity order

1. Extract fourth-order field equations:

```
In[]:= eqns4 = PPN[MetEq, 4][-LI[0], -LI[0]] /. meteqvru;
```

Equations at the fourth velocity order

1. Extract fourth-order field equations:

```
In[]:= eqns4 = PPN[MetEq, 4][-LI[0], -LI[0]] /. meteqvru;
```

2. We find that it takes the following form:

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

Equations at the fourth velocity order

1. Extract fourth-order field equations:

```
In[]:= eqns4 = PPN[MetEq, 4][-LI[0], -LI[0]] /. meteqvru;
```

2. We find that it takes the following form:

$$\overset{4}{\mathcal{E}}_{00} = -\kappa^2\rho v^2 - \kappa^2\frac{\omega(\Psi)+2}{2\omega(\Psi)+3}\rho\Pi - 3\kappa^2\frac{\omega(\Psi)+3}{2\omega(\Psi)+3}p$$

$$-\frac{\Psi}{4}\left(2\triangle\overset{4}{g}_{00} - 4\overset{3}{g}_{0a,0a} + 2\overset{2}{g}_{aa,00} + \overset{2}{g}_{00,a}\overset{2}{g}_{00,a} + \overset{2}{g}_{00,a}\overset{2}{g}_{bb,a} - 2\overset{2}{g}_{00,a}\overset{2}{g}_{ab,b} - \overset{2}{g}_{00,ab}\overset{2}{g}_{ab}\right)$$

$$-\frac{\omega'(\Psi)}{4\omega(\Psi)+6}\overset{2}{\psi}_{,a}\overset{2}{\psi}_{,a} + \frac{\kappa^2\omega'(\Psi)}{(2\omega(\Psi)+3)^2}\rho\overset{2}{\psi} + \kappa^2\frac{\omega(\Psi)+2}{2\omega(\Psi)+3}\rho\overset{2}{g}_{00} - \frac{1}{2}\overset{2}{\psi}_{,a}\overset{2}{g}_{00,a} - \frac{1}{2}\overset{2}{\psi}\triangle\overset{2}{g}_{00} - \overset{2}{\psi}_{,00}$$

3. Ansatz for fourth-order metric component:

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

Insert ansatz into fourth-order field equations

```
In[]:= eqns4 /. ans4ru /. sol2ru /. sol3ru;
In[]:= Expand[%];
In[]:= ContractMetric[%, OverDerivatives → True,
   AllowUpperDerivatives → True];
In[]:= PotentialVToU[%];
In[]:= PotentialWToU[%];
In[]:= PotentialToSource[%];
In[]:= ToCanonical[%];
In[]:= SortPDs[%];
In[]:= Expand[%];
In[]:= eqnsa4 = Simplify[ScreenDollarIndices[%]];
```

Extract equations for constant coefficients

```
In[]:= eq1 = Simplify[Coefficient[eqnsa4,
    Pressure[]]];
In[]:= eq2 = Simplify[Coefficient[eqnsa4,
    Density[] * InternalEnergy[]]];
In[]:= eq3 = Simplify[Coefficient[eqnsa4,
    Density[] * PotentialU[]]];
In[]:= eq4 = Simplify[Coefficient[eqnsa4,
    ParamD[TimePar, TimePar][PotentialU[]]]];
In[]:= eq5 = Simplify[Coefficient[eqnsa4,
    Density[] * Velocity[-T3a] * Velocity[T3a]]];
In[]:= eq6 = Simplify[Coefficient[eqnsa4,
    PD[-T3a][PotentialU[]] * PD[T3a][PotentialU[]]]];
```

Check and solve decomposed equations

1. Check that fourth-order field equations are fully decomposed:

```
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 -
   eqnsa4]
Out[]= 0
```

Check and solve decomposed equations

1. Check that fourth-order field equations are fully decomposed:

```
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 -
    eqnsa4]
Out[]= 0
```

2. Solve equations for constant coefficients:

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

Check and insert solution for constant coefficients

1. Check that solution indeed solved fourth-order field equations:

```
In[]:= Simplify[eqnsa4 /. sola4]
Out[]= 0
```

Check and insert solution for constant coefficients

1. Check that solution indeed solved fourth-order field equations:

```
In[]:= Simplify[eqnsa4 /. sola4]
Out[]= 0
```

2. Using coefficient a_0, obtain complete third order solution with $a_{5,6}$:

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

Check and insert solution for constant coefficients

1. Check that solution indeed solved fourth-order field equations:

```
In[]:= Simplify[eqnsa4 /. sola4]
Out[]= 0
```

2. Using coefficient a_0, obtain complete third order solution with $a_{5,6}$:

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

3. Use remaining coefficients to obtain solution for $\overset{4}{g}_{00}$:

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

Check fourth-order field equations

```
In[]:= eqns4 /. 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
```

Collect metric components

1. Metric components needed to solve for PPN parameters:

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

Collect metric components

1. Metric components needed to solve for PPN parameters:

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

2. Metric components with solution we have determined:

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

Collect metric components

1. Metric components needed to solve for PPN parameters:

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

2. Metric components with solution we have determined:

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

3. We will compare this to the standard PPN metric:

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

Newtonian gravitational constant

1. Compare second-order component $\overset{2}{g}_{00}$ with standard normalization:

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

Newtonian gravitational constant

1. Compare second-order component $\overset{2}{g}_{00}$ with standard normalization:

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

2. Equation takes the form:

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

Newtonian gravitational constant

1. Compare second-order component $\overset{2}{g}_{00}$ with standard normalization:

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

2. Equation takes the form:

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

3. 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];
```

Newtonian gravitational constant

1. Compare second-order component $\overset{2}{g}_{00}$ with standard normalization:

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

2. Equation takes the form:

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

3. 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];
```

4. This yields the solution:

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

PPN potentials and their coefficients

1. Final equations appear as coefficients in front of potentials:

```
In[]:= pots = {PotentialU[] BkgMetricS3[-T3a, -T3b],
   PotentialV[-T3a], PotentialW[-T3a],
   PotentialA[], PotentialU[]^2, PotentialPhiW[],
   PotentialPhi1[], PotentialPhi2[],
   PotentialPhi3[], PotentialPhi4[]};
```

PPN potentials and their coefficients

1. Final equations appear as coefficients in front of potentials:

```
In[]:= pots = {PotentialU[] BkgMetricS3[-T3a, -T3b],
   PotentialV[-T3a], PotentialW[-T3a],
   PotentialA[], PotentialU[]^2, PotentialPhiW[],
   PotentialPhi1[], PotentialPhi2[],
   PotentialPhi3[], PotentialPhi4[]};
```

2. Read off equations as coefficients of PPN potentials:

```
In[]:= eqns = DeleteCases[Flatten[Simplify[
   Outer[Coefficient, pareqns, pots]]], 0];
```

Solve for PPN parameters

1. PPN parameters to be solved form:

```
In[]:= pars = {ParameterBeta, ParameterGamma, ParameterXi,
    ParameterAlpha1, ParameterAlpha2, ParameterAlpha3,
    ParameterZeta1, ParameterZeta2,
    ParameterZeta3, ParameterZeta4};
```

Solve for PPN parameters

1. PPN parameters to be solved form:

```
In[]:= pars = {ParameterBeta, ParameterGamma, ParameterXi,
    ParameterAlpha1, ParameterAlpha2, ParameterAlpha3,
    ParameterZeta1, ParameterZeta2,
    ParameterZeta3, ParameterZeta4};
```

2. Solve equations for PPN parameters:

```
In[]:= parsol = FullSimplify[Solve[
    # == 0 & /@ eqns, pars][[1]]];
```

Solve for PPN parameters

1. PPN parameters to be solved form:

```
In[]:= pars = {ParameterBeta, ParameterGamma, ParameterXi,
    ParameterAlpha1, ParameterAlpha2, ParameterAlpha3,
    ParameterZeta1, ParameterZeta2,
    ParameterZeta3, ParameterZeta4};
```

2. Solve equations for PPN parameters:

```
In[]:= parsol = FullSimplify[Solve[
    # == 0 & /@ eqns, pars][[1]]];
```

3. This finally yields the solution:

$$\gamma = \frac{\omega(\Psi) + 1}{\omega(\Psi) + 2}, \quad \beta = 1 + \frac{\Psi\omega'(\Psi)}{4(2\omega(\Psi) + 3)(\omega(\Psi) + 2)^2},$$

$$\alpha_1 = \alpha_2 = \alpha_3 = \zeta_1 = \zeta_2 = \zeta_3 = \zeta_4 = \xi = 0\,.$$

Solve for PPN parameters

1. PPN parameters to be solved form:

```
In[]:= pars = {ParameterBeta, ParameterGamma, ParameterXi,
    ParameterAlpha1, ParameterAlpha2, ParameterAlpha3,
    ParameterZeta1, ParameterZeta2,
    ParameterZeta3, ParameterZeta4};
```

2. Solve equations for PPN parameters:

```
In[]:= parsol = FullSimplify[Solve[
    # == 0 & /@ eqns, pars][[1]]];
```

3. This finally yields the solution:

$$\gamma = \frac{\omega(\Psi) + 1}{\omega(\Psi) + 2}, \quad \beta = 1 + \frac{\Psi\omega'(\Psi)}{4(2\omega(\Psi) + 3)(\omega(\Psi) + 2)^2},$$

$$\alpha_1 = \alpha_2 = \alpha_3 = \zeta_1 = \zeta_2 = \zeta_3 = \zeta_4 = \xi = 0\,.$$

✓ Obtain well-known PPN parameters for massless scalar-tensor gravity [Nordtvedt '70].

Outline

Summary

- Parametrized post-Newtonian formalism:
    - Weak-field approximation of metric gravity theories.
    - Characterizes gravity theories by 10 (constant) parameters.
    - Parameters closely related to solar system observations.
    - Can also be applied to teleparallel and other gravity theories.

Summary

- Parametrized post-Newtonian formalism:
  - Weak-field approximation of metric gravity theories.
  - Characterizes gravity theories by 10 (constant) parameters.
  - Parameters closely related to solar system observations.
  - Can also be applied to teleparallel and other gravity theories.
- *xAct* for Mathematica:
  - Versatile tensor algebra package with numerous helpful functions.
  - Built upon powerful computer algebra system.
  - Easily extendable with new functions specific to physical applications

Summary

- Parametrized post-Newtonian formalism:
  - Weak-field approximation of metric gravity theories.
  - Characterizes gravity theories by 10 (constant) parameters.
  - Parameters closely related to solar system observations.
  - Can also be applied to teleparallel and other gravity theories.
- *xAct* for Mathematica:
  - Versatile tensor algebra package with numerous helpful functions.
  - Built upon powerful computer algebra system.
  - Easily extendable with new functions specific to physical applications
- *xPPN*: post-Newtonian formalism implemented in *xAct*:
  - Automatic rules for $3 + 1$ split and perturbative expansion of tensor fields.
  - Numerous pre-defined objects to represent fields in PPN formalism.
  - Numerous pre-defined rules implementing relations and transformations.
  - ⇒ Greatly simplifies task of solving post-Newtonian field equations.

Outlook

- Extend package by further functions and fields:
    - More general connections to study Poincare / metric-affine gravity theories.
    - Allow for additional metric tensors / tetrads.
    - Include more general PPN potentials for massive / higher derivative gravity.
    - Make use of gauge-invariant PPN formalism [MH '19].

Outlook

- Extend package by further functions and fields:
  - More general connections to study Poincare / metric-affine gravity theories.
  - Allow for additional metric tensors / tetrads.
  - Include more general PPN potentials for massive / higher derivative gravity.
  - Make use of gauge-invariant PPN formalism [MH '19].
- Apply calculations to complicated gravity theories:
  - Bimetric and multimetric gravity theories.
  - Multi-scalar Horndeski generalizations.
  - Theories involving generalized Proca fields.
  - Extensions based on metric-affine geometry.
  - Extensions of teleparallel and symmetric teleparallel gravity.

Outlook

- Extend package by further functions and fields:
  - More general connections to study Poincare / metric-affine gravity theories.
  - Allow for additional metric tensors / tetrads.
  - Include more general PPN potentials for massive / higher derivative gravity.
  - Make use of gauge-invariant PPN formalism [MH '19].
- Apply calculations to complicated gravity theories:
  - Bimetric and multimetric gravity theories.
  - Multi-scalar Horndeski generalizations.
  - Theories involving generalized Proca fields.
  - Extensions based on metric-affine geometry.
  - Extensions of teleparallel and symmetric teleparallel gravity.

→ `https://github.com/xenos1984/xPPN`