Part II: GWs in the lattice

Recap of lattice discretization (I)

We work in a real periodic lattice with spacing δx

$$\boldsymbol{n} = (n_1, n_2, n_3), \qquad n_i = 0, 1, ..., N-1$$

Recap of lattice discretization (I)

We work in a real periodic lattice with spacing δx

$$\boldsymbol{n} = (n_1, n_2, n_3), \qquad n_i = 0, 1, ..., N-1$$

Fourier modes also live in a periodic lattice with spacing $k_{\rm IR} = 2\pi/L$

$$\tilde{\boldsymbol{n}} = (\tilde{n}_1, \tilde{n}_2, \tilde{n}_3), \qquad \tilde{n}_i = -\frac{N}{2} + 1, ..., -1, 0, 1, ..., \frac{N}{2}$$

Recap of lattice discretization (I)

We work in a real periodic lattice with spacing δx

$$\boldsymbol{n} = (n_1, n_2, n_3), \qquad n_i = 0, 1, ..., N-1$$

Fourier modes also live in a periodic lattice with spacing $k_{\rm IR} = 2\pi/L$

$$\tilde{\boldsymbol{n}} = (\tilde{n}_1, \tilde{n}_2, \tilde{n}_3), \qquad \tilde{n}_i = -\frac{N}{2} + 1, ..., -1, 0, 1, ..., \frac{N}{2}$$

Related by discrete Fourier transform

$$f(\boldsymbol{n}) = \frac{1}{N^3} \sum_{\tilde{\boldsymbol{n}}} e^{\frac{2\pi i}{N} \tilde{\boldsymbol{n}} \boldsymbol{n}} f(\tilde{\boldsymbol{n}}) \qquad f(\tilde{\boldsymbol{n}}) = \sum_{\boldsymbol{n}} e^{-\frac{2\pi i}{N} \tilde{\boldsymbol{n}} \boldsymbol{n}} f(\boldsymbol{n})$$

Recap of lattice discretization (II)

Recap of lattice discretization (II)

We define a lattice momentum

$$[\nabla_i f](\tilde{\boldsymbol{n}}) = -i\boldsymbol{k}_{\mathrm{L}}(\tilde{\boldsymbol{n}})f(\tilde{\boldsymbol{n}})$$

Recap of lattice discretization (II)

We define a lattice momentum

$$[\nabla_i f](\tilde{\boldsymbol{n}}) = -i\boldsymbol{k}_{\mathrm{L}}(\tilde{\boldsymbol{n}}) f(\tilde{\boldsymbol{n}})$$

Different derivative discretizations

$$[\nabla_i^0 f](\boldsymbol{n}) = \frac{f(\boldsymbol{n}+\hat{i}) - f(\boldsymbol{n}-\hat{i})}{2\delta x} \quad \longrightarrow \quad k_{\mathrm{L},i}^0 = \frac{\sin(2\pi\tilde{n}_i/N)}{\delta x}$$

$$[\nabla_i^\pm f](\boldsymbol{n}) = \frac{\pm f(\boldsymbol{n}\pm\hat{i}) \mp f(\boldsymbol{n})}{\delta x} \quad \longrightarrow \quad k_{\mathrm{L},i}^\pm = 2e^{\mp i\pi\tilde{n}_i/N}\frac{\sin(\pi\tilde{n}_i/N)}{\delta x}$$

GWs in the lattice

GWs in the lattice

We work with six **unphysical** real degrees of freedom

$$\ddot{u}_{ij}(\boldsymbol{n}) + 3H\dot{u}_{ij}(\boldsymbol{n}) - \frac{\nabla_{\rm L}^2}{a^2}u_{ij}(\boldsymbol{n}) = \frac{2}{m_{\rm Pl}^2 a^2}\Pi_{ij}(\boldsymbol{n})$$

GWs in the lattice

We work with six **unphysical** real degrees of freedom

$$\ddot{u}_{ij}(\boldsymbol{n}) + 3H\dot{u}_{ij}(\boldsymbol{n}) - \frac{\nabla_{\rm L}^2}{a^2} u_{ij}(\boldsymbol{n}) = \frac{2}{m_{\rm Pl}^2 a^2}\Pi_{ij}(\boldsymbol{n})$$

Physical fields projected **only** when measuring

$$h_{ij}(\tilde{\boldsymbol{n}}, t) = \Lambda^{\rm L}_{ij,kl}(\tilde{\boldsymbol{n}}) u_{kl}(\tilde{\boldsymbol{n}}, t)$$

GWs in the lattice

We work with six **unphysical** real degrees of freedom

$$\ddot{u}_{ij}(\boldsymbol{n}) + 3H\dot{u}_{ij}(\boldsymbol{n}) - \frac{\nabla_{\rm L}^2}{a^2}u_{ij}(\boldsymbol{n}) = \frac{2}{m_{\rm Pl}^2 a^2}\Pi_{ij}(\boldsymbol{n})$$

Physical fields projected **only** when measuring

$$h_{ij}(\tilde{\boldsymbol{n}}, t) = \Lambda^{\rm L}_{ij,kl}(\tilde{\boldsymbol{n}})u_{kl}(\tilde{\boldsymbol{n}}, t)$$

Need to choose a particular lattice derivative

$$\Lambda^{\rm L}_{ij,kl} = P_{il}P_{jm} - \frac{1}{2}P_{ij}P_{lm} \qquad P_{ij} = \delta_{ij} - \frac{k_{{\rm L},i}k_{{\rm L},j}}{k_{\rm L}^2}$$

Ensures $h_{ii} = 0$ and $\nabla_{{\rm L},i}h_{ij} = 0$ [Figueroa, García-Bellido, Ranjantie, (2011), 1110.0337]

Properties of the TT projector

For the **forward/backward** derivatives,

$$\sum_i k_{\mathrm{L},i} P_{ij} = 0 \qquad \sum_i k^*_{\mathrm{L},i} P_{ij} \neq 0$$

$$\sum_j k_{\mathrm{L},j} P_{ij} \neq 0 \qquad \sum_j k^*_{\mathrm{L},j} P_{ij} = 0$$

$$P^*_{ij} = P_{ji} \qquad P_{ij}(-\tilde{\boldsymbol{n}}) = P_{ij}(\tilde{\boldsymbol{n}})$$

$$P_{ij} P_{jk} = P_{ik} \qquad P_{ij} P_{ki} \neq P_{ik}$$

[Figueroa, García-Bellido, Ranjantie, (2011), 1110.0337]

Properties of the TT projector

For the **forward/backward** derivatives,

$$\sum_i k_{\mathrm{L},i} P_{ij} = 0 \qquad \sum_i k^*_{\mathrm{L},i} P_{ij} \neq 0$$

$$\sum_j k_{\mathrm{L},j} P_{ij} \neq 0 \qquad \sum_j k^*_{\mathrm{L},j} P_{ij} = 0$$

$$\boxed{P^*_{ij} = P_{ji}} \qquad P_{ij}(-\tilde{\boldsymbol{n}}) = P_{ij}(\tilde{\boldsymbol{n}})$$

$$P_{ij}P_{jk} = P_{ik} \qquad P_{ij}P_{ki} \neq P_{ik}$$

[Figueroa, García-Bellido, Ranjantie, (2011), 1110.0337]

Properties of the TT projector

For the **forward/backward** derivatives,

$$\sum_i k_{\mathrm{L},i} P_{ij} = 0 \qquad \sum_i k^*_{\mathrm{L},i} P_{ij} \neq 0$$

$$\sum_j k_{\mathrm{L},j} P_{ij} \neq 0 \qquad \sum_j k^*_{\mathrm{L},j} P_{ij} = 0$$

$$\boxed{P^*_{ij} = P_{ji}} \qquad P_{ij}(-\tilde{\boldsymbol{n}}) = P_{ij}(\tilde{\boldsymbol{n}})$$

$$\boxed{P_{ij} P_{jk} = P_{ik}} \qquad P_{ij} P_{ki} \neq P_{ik}$$

[Figueroa, García-Bellido, Ranjantie, (2011), 1110.0337]

Properties of the TT projector

For the **forward/backward** derivatives,

$$\sum_i k_{\mathrm{L},i} P_{ij} = 0 \qquad \sum_i k^*_{\mathrm{L},i} P_{ij} \neq 0$$

$$\sum_j k_{\mathrm{L},j} P_{ij} \neq 0 \qquad \sum_j k^*_{\mathrm{L},j} P_{ij} = 0$$

$$\boxed{P^*_{ij} = P_{ji}} \qquad P_{ij}(-\tilde{\boldsymbol{n}}) = P_{ij}(\tilde{\boldsymbol{n}})$$

$$\boxed{P_{ij} P_{jk} = P_{ik}} \qquad P_{ij} P_{ki} \neq P_{ik}$$

Similar set of properties for real projector

[Figueroa, García-Bellido, Ranjantie, (2011), 1110.0337]

Properties of the TT projector

For the **forward/backward** derivatives,

$$\sum_i k_{\mathrm{L},i} P_{ij} = 0 \qquad \sum_i k^*_{\mathrm{L},i} P_{ij} \neq 0$$

$$\sum_j k_{\mathrm{L},j} P_{ij} \neq 0 \qquad \sum_j k^*_{\mathrm{L},j} P_{ij} = 0$$

$$\boxed{P^*_{ij} = P_{ji}} \qquad P_{ij}(-\tilde{\boldsymbol{n}}) = P_{ij}(\tilde{\boldsymbol{n}})$$

$$\boxed{P_{ij} P_{jk} = P_{ik}} \qquad P_{ij} P_{ki} \neq P_{ik}$$

Similar set of properties for real projector

[Figueroa, García-Bellido, Ranjantie, (2011), 1110.0337]

GW energy density power spectrum

GW energy density power spectrum

We define the GW energy density as a volume average

$$\rho_{\rm GW}(t) \quad = \frac{1}{32\pi G}\langle \dot{h}_{ij}\dot{h}_{ij}\rangle_V$$

GW energy density power spectrum

We define the GW energy density as a volume average

$$\rho_{\rm GW}(t) \quad = \frac{1}{32\pi G}\langle \dot{h}_{ij}\dot{h}_{ij}\rangle_V = \frac{1}{32\pi G N^3}\sum_{\boldsymbol{n}} \dot{h}_{ij}(\boldsymbol{n},t)\dot{h}_{ij}(\boldsymbol{n},t)$$

GW energy density power spectrum

We define the GW energy density as a volume average

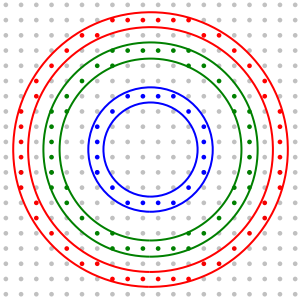
$$\begin{aligned}\rho_{\rm GW}(t) &= \frac{1}{32\pi G}\langle \dot{h}_{ij}\dot{h}_{ij}\rangle_V = \frac{1}{32\pi G N^3}\sum_{\boldsymbol{n}} \dot{h}_{ij}(\boldsymbol{n},t)\dot{h}_{ij}(\boldsymbol{n},t)\\ &= \frac{1}{32\pi G N^6}\sum_{\tilde{\boldsymbol{n}}} \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t)\end{aligned}$$

Fourier transform
Parseval's theorem

GW energy density power spectrum

We define the GW energy density as a volume average

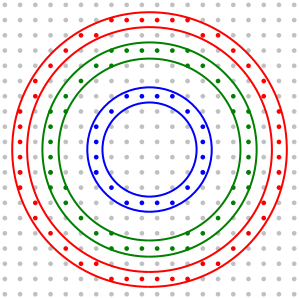
$$
\begin{aligned}
\rho_{\mathrm{GW}}(t) &= \frac{1}{32\pi G}\langle \dot h_{ij}\dot h_{ij}\rangle_V = \frac{1}{32\pi G N^3}\sum_{\boldsymbol{n}} \dot h_{ij}(\boldsymbol{n},t)\dot h_{ij}(\boldsymbol{n},t)\\
&= \frac{1}{32\pi G N^6}\sum_{\tilde{\boldsymbol{n}}} \dot h_{ij}(\tilde{\boldsymbol{n}},t)\dot h^*_{ij}(\tilde{\boldsymbol{n}},t)\\
&= \frac{1}{32\pi G N^6}\sum_{\tilde n}\sum_{R(\tilde n)} \dot h_{ij}(\tilde{\boldsymbol{n}},t)\dot h^*_{ij}(\tilde{\boldsymbol{n}},t)
\end{aligned}
$$



GW energy density power spectrum

We define the GW energy density as a volume average

$$
\begin{aligned}
\rho_{\rm GW}(t) &= \frac{1}{32\pi G}\langle \dot{h}_{ij}\dot{h}_{ij}\rangle_V = \frac{1}{32\pi G N^3}\sum_{\boldsymbol{n}} \dot{h}_{ij}(\boldsymbol{n},t)\dot{h}_{ij}(\boldsymbol{n},t)\\
&= \frac{1}{32\pi G N^6}\sum_{\tilde{\boldsymbol{n}}} \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t)\\
&= \frac{1}{32\pi G N^6}\sum_{\tilde{n}}\sum_{R(\tilde{n})} \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t)
\end{aligned}
$$

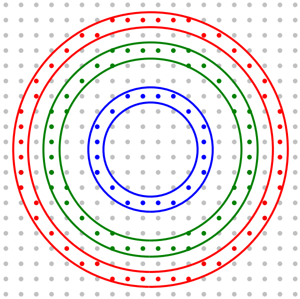


Multiplicity of shells
$\approx 4\pi\tilde{n}^2$

GW energy density power spectrum

We define the GW energy density as a volume average

$$
\begin{aligned}
\rho_{\rm GW}(t) \quad &= \frac{1}{32\pi G}\langle \dot{h}_{ij}\dot{h}_{ij}\rangle_V = \frac{1}{32\pi G N^3}\sum_{\boldsymbol{n}} \dot{h}_{ij}(\boldsymbol{n},t)\dot{h}_{ij}(\boldsymbol{n},t) \\
&= \frac{1}{32\pi G N^6}\sum_{\tilde{\boldsymbol{n}}} \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t) \\
&= \frac{1}{32\pi G N^6}\sum_{\tilde{n}} 4\pi\tilde{n}^2\langle \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t)\rangle_{R(\tilde{n})}
\end{aligned}
$$



Multiplicity of shells
$\approx 4\pi\tilde{n}^2$

GW energy density power spectrum

We define the GW energy density as a volume average

$$\begin{aligned}
\rho_{\rm GW}(t) &= \frac{1}{32\pi G}\langle \dot{h}_{ij}\dot{h}_{ij}\rangle_V = \frac{1}{32\pi G N^3}\sum_{\boldsymbol{n}} \dot{h}_{ij}(\boldsymbol{n},t)\dot{h}_{ij}(\boldsymbol{n},t) \\
&= \frac{1}{32\pi G N^6}\sum_{\tilde{\boldsymbol{n}}} \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t) \\
&= \frac{1}{32\pi G N^6}\sum_{\tilde{n}} 4\pi\tilde{n}^2\langle \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t)\rangle_{R(\tilde{n})} \\
&= \sum_{\tilde{n}}\left[\frac{\delta x^6}{(4\pi)^3 G L^3}k^3(\tilde{n})\langle \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t)\rangle_{R(\tilde{n})}\right]\Delta\log k
\end{aligned}$$

$$\Delta\log k \downarrow k_{\rm IR}/k(\tilde{n})$$

GW energy density power spectrum

We define the GW energy density as a volume average

$$
\begin{aligned}
\rho_{\rm GW}(t) &= \frac{1}{32\pi G}\langle \dot{h}_{ij}\dot{h}_{ij}\rangle_V = \frac{1}{32\pi G N^3}\sum_{\boldsymbol{n}} \dot{h}_{ij}(\boldsymbol{n},t)\dot{h}_{ij}(\boldsymbol{n},t) \\
&= \frac{1}{32\pi G N^6}\sum_{\tilde{\boldsymbol{n}}} \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t) \\
&= \frac{1}{32\pi G N^6}\sum_{\tilde{n}} 4\pi\tilde{n}^2\langle \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t)\rangle_{R(\tilde{n})} \\
&= \sum_{\tilde{n}} \underbrace{\left[\frac{\delta x^6}{(4\pi)^3 G L^3} k^3(\tilde{n})\langle \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t)\rangle_{R(\tilde{n})}\right]}_{\frac{\mathrm{d}\rho_{\rm GW}}{\mathrm{d}\log k}} \underset{k_{\rm IR}/k(\tilde{n})}{\underset{\downarrow}{\Delta\log k}}
\end{aligned}
$$

GW energy density power spectrum

We define the GW energy density as a volume average

$$
\begin{aligned}
\rho_{\rm GW}(t) &= \frac{1}{32\pi G}\langle \dot{h}_{ij}\dot{h}_{ij}\rangle_V = \frac{1}{32\pi G N^3}\sum_{\boldsymbol{n}} \dot{h}_{ij}(\boldsymbol{n},t)\dot{h}_{ij}(\boldsymbol{n},t) \\
&= \frac{1}{32\pi G N^6}\sum_{\tilde{\boldsymbol{n}}} \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t) \\
&= \frac{1}{32\pi G N^6}\sum_{\tilde{n}} 4\pi\tilde{n}^2\langle \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t)\rangle_{R(\tilde{n})} \\
&= \sum_{\tilde{n}} \underbrace{\left[\frac{\delta x^6}{(4\pi)^3 G L^3}k^3(\tilde{n})\langle \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t)\rangle_{R(\tilde{n})}\right]}_{\frac{d\rho_{\rm GW}}{d\log k}} \underbrace{\Delta\log k}_{k_{\rm IR}/k(\tilde{n})}
\end{aligned}
$$

Using the TT projector

$$
\dot{h}_{ij}\dot{h}^*_{ij} = \Lambda_{ij,kl}\dot{u}_{kl}\Lambda^*_{ij,mp}\dot{u}^*_{mp} = \mathrm{Tr}[PuPu^*] - \frac{1}{2}\mathrm{Tr}[Pu]\mathrm{Tr}[Pu^*]
$$

We **only** project when we want to measure

GW energy density power spectrum

We define the GW energy density as a volume average

$$
\begin{aligned}
\rho_{\mathrm{GW}}(t) &= \frac{1}{32\pi G}\langle \dot{h}_{ij}\dot{h}_{ij}\rangle_V = \frac{1}{32\pi G N^3}\sum_{\boldsymbol{n}} \dot{h}_{ij}(\boldsymbol{n},t)\dot{h}_{ij}(\boldsymbol{n},t) \\
&= \frac{1}{32\pi G N^6}\sum_{\tilde{\boldsymbol{n}}} \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t) \\
&= \frac{1}{32\pi G N^6}\sum_{\tilde{n}} 4\pi\tilde{n}^2 \langle \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t)\rangle_{R(\tilde{n})} \\
&= \sum_{\tilde{n}} \underbrace{\left[\frac{\delta x^6}{(4\pi)^3 G L^3} k^3(\tilde{n}) \langle \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t)\rangle_{R(\tilde{n})}\right]}_{\frac{\mathrm{d}\rho_{\mathrm{GW}}}{\mathrm{d}\log k}} \underset{\substack{\big\downarrow \\ k_{\mathrm{IR}}/k(\tilde{n})}}{\Delta \log k}
\end{aligned}
$$

In general we compute the **normalized** energy density power spectrum

$$
\Omega_{\mathrm{GW}} = \frac{1}{\rho_{\mathrm{c}}}\frac{\mathrm{d}\rho_{\mathrm{GW}}}{\mathrm{d}\log k}
$$

Part III: Gravitational waves in $\mathcal{C}$osmo$\mathcal{L}$attice

J. Baeza-Ballesteros, D. G. Figueroa, A. Florio and N. Loayza

Equations of motion

In $\mathcal{C}$osmo$\mathcal{L}$attice, we work with program variables

$$\tilde{\eta} = \omega_* a^{-\alpha} t \qquad \tilde{x}^i = \omega_* x^i \qquad \kappa = k/\omega_* \qquad \tilde{\phi} = \phi/f_*$$

Equations of motion

In $\mathcal{C}$osmo$\mathcal{L}$attice, we work with program variables

$$\tilde{\eta} = \omega_* a^{-\alpha} t \qquad \tilde{x}^i = \omega_* x^i \qquad \kappa = k/\omega_* \qquad \tilde{\phi} = \phi/f_*$$

We evolve the field using a conjugate momenta, $(\pi_u)_{ij} = a^{3-\alpha} u'_{ij}$

$$\begin{cases} u'_{ij} &= a^{\alpha-3}(\pi_u)_{ij} \\ (\pi_u)'_{ij} &= a^{1+\alpha}\tilde{\nabla}^2 u_{ij} + 2a^{1+\alpha}\left(\dfrac{f_*}{m_{\rm Pl}}\right)^2 \tilde{\Pi}_{ij} \end{cases}$$

Equations of motion

We evolve the field using a conjugate momenta, $(\pi_u)_{ij} = a^{3-\alpha} u'_{ij}$

$$\begin{cases} u'_{ij} &= a^{\alpha-3}(\pi_u)_{ij} \\ (\pi_u)'_{ij} &= a^{1+\alpha}\tilde{\nabla}^2 u_{ij} + 2a^{1+\alpha}\left(\frac{f_*}{m_{\rm Pl}}\right)^2 \tilde{\Pi}_{ij} \end{cases}$$

Equations of motion

We evolve the field using a conjugate momenta, $(\pi_u)_{ij} = a^{3-\alpha} u'_{ij}$

$$\begin{cases} u'_{ij} &=& a^{\alpha-3}(\pi_u)_{ij} \\ (\pi_u)'_{ij} &=& a^{1+\alpha}\tilde{\nabla}^2 u_{ij} + 2a^{1+\alpha}\left(\frac{f_*}{m_{\rm Pl}}\right)^2 \tilde{\Pi}_{ij} \end{cases}$$

$\mathcal{C}$osmo$\mathcal{L}$attice is prepared to simulate GWs sourced by scalar fields

$$\tilde{\Pi}_{ij} = \sum_a \tilde{\partial}_i \tilde{\phi}_a \tilde{\partial}_j \tilde{\phi}_a$$

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice

Please, open the following files:

```
src/model/parameter-files/lphi4.in

src/include/CosmoInterface/evolvers/leapfrog.h

src/include/CosmoInterface/evolvers/kernels/gwskernels.h

src/include/CosmoInterface/definitions/PITensor.h

src/include/CosmoInterface/definitions/GWsProjector.h

src/include/CosmoInterface/measurements/gwspowerspectrum.h
```

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: input parameters

src/model/parameter-files/lphi4.in

```
#Times
tOutputFreq = 1
tOutputInfreq = 1
tMax = 300

#Spectra options
PS_type = 1
PS_version = 1

#GWs
GWprojectorType = 2
withGWs=true

#IC
kCutOff = 4
initial_amplitudes = 5.6964e18 0  # homogeneous amplitudes in GeV
initial_momenta = -4.86735e30 0   # homogeneous amplitudes in GeV2
```

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: input parameters

src/model/parameter-files/lphi4.in

```
#Times
tOutputFreq = 1
tOutputInfreq = 1
tMax = 300

#Spectra options
PS_type = 1
PS_version = 1

#GWs
GWprojectorType = 2
withGWs=true

#IC
kCutOff = 4
initial_amplitudes = 5.6964e18 0  # homogeneous amplitudes in GeV
initial_momenta = -4.86735e30 0   # homogeneous amplitudes in GeV2
```

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: input parameters

`src/model/parameter-files/lphi4.in`

```
#Times
tOutputFreq = 1
tOutputInfreq = 1
tMax = 300

#Spectra options
PS_type = 1
PS_version = 1

#GWs
GWprojectorType = 2
withGWs=true

#IC
kCutOff = 4
initial_amplitudes = 5.6964e18 0  # homogeneous amplitudes in GeV
initial_momenta = -4.86735e30 0   # homogeneous amplitudes in GeV2
```

The GWs module is controlled by two parameters:

1. `withGWs`: boolean used to turn On/Off GWs

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: input parameters

src/model/parameter-files/lphi4.in

```
#Times
tOutputFreq = 1
tOutputInfreq = 1
tMax = 300

#Spectra options
PS_type = 1
PS_version = 1

#GWs
GWprojectorType = 2
withGWs=true

#IC
kCutOff = 4
initial_amplitudes = 5.6964e18 0  # homogeneous amplitudes in GeV
initial_momenta = -4.86735e30 0   # homogeneous amplitudes in GeV2
```

The GWs module is controlled by two parameters:

1. `withGWs`: boolean used to turn On/Off GWs
2. `GWprojectorType`: choose between different projectors
   - `GWprojectorType = 1`: neutral derivative
   - `GWprojectorType = 2`: forward derivative (default)
   - `GWprojectorType = 3`: backward derivative

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: output results

We get two different output files

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: output results

We get two different output files

1. `spectra_gws.txt`

κ	Ω_{GW}	Multiplicity
0.4	3.455976932183128e-29	13
0.8	1.135499488356703e-28	37
1.2	1.653950018181714e-28	57
⋮	⋮	⋮

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: output results

We get two different output files

1. `spectra_gws.txt`

$$\Omega_{\rm GW} \rightarrow \frac{1}{\tilde{\rho}_{\rm tot}} \frac{d\tilde{\rho}_{\rm GW}}{d\log k}$$

κ	$\Omega_{\rm GW}$	Multiplicity
0.4	3.455976932183128e-29	13
0.8	1.135499488356703e-28	37
1.2	1.653950018181714e-28	57
$\vdots$	$\vdots$	$\vdots$

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: output results

We get two different output files

1. `spectra_gws.txt`

$$\Omega_{\rm GW} \rightarrow \frac{1}{\tilde{\rho}_{\rm tot}}\frac{d\tilde{\rho}_{\rm GW}}{d\log k}$$

κ	$\Omega_{\rm GW}$	Multiplicity
0.4	3.455976932183128e-29	13
0.8	1.135499488356703e-28	37
1.2	1.653950018181714e-28	57
⋮	⋮	⋮

2. `energy_gws.txt`

$\tilde{\eta}$	$\tilde{\rho}_{\rm GW}/\tilde{\rho}_{\rm tot}$	$\tilde{\rho}_{\rm GW} = \rho_{\rm GW}/\omega_*^2 f_*^2$
0	0	0
1	1.26699487198562e-27	5.29787700163311e-29
2	2.75524033399837e-27	1.24679240886064e-29
⋮	⋮	⋮

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: equation of motion

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: equation of motion

src/include/CosmoInterface/evolvers/leapfrog.h

```
if (model.fldGWs != nullptr) kickGWs(model, weight);
```

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: equation of motion

src/include/CosmoInterface/evolvers/leapfrog.h

```
if (model.fldGWs != nullptr) kickGWs(model, weight);
```

```
template<class Model>
void kickGWs(Model& model, T w) {
    ForLoop(n, 0, Model::NGWs - 1,
            (*model.piGWs)(n) += (w * model.dt) *
            ↪  GWsKernels::get(model,n);
    );
}
```

$$\boxed{(\pi_u)'_{ij} = \mathcal{K}_{ij}[u_{ij}, \{\tilde{\phi}\}, a]}$$

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: equation of motion

`src/include/CosmoInterface/evolvers/leapfrog.h`

```
if (model.fldGWs != nullptr) kickGWs(model, weight);
```

```
template<class Model>
void kickGWs(Model& model, T w) {
    ForLoop(n, 0, Model::NGWs - 1,
            (*model.piGWs)(n) += (w * model.dt) *
            ↪  GWsKernels::get(model,n);
    );
}
```

$$\boxed{(\pi_u)'_{ij} = \mathcal{K}_{ij}[u_{ij}, \{\tilde{\phi}\}, a]}$$

```
if (model.fldGWs != nullptr) driftGWs(model);
```

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: equation of motion

`src/include/CosmoInterface/evolvers/leapfrog.h`

```
if (model.fldGWs != nullptr) kickGWs(model, weight);
```

```
template<class Model>
void kickGWs(Model& model, T w) {
    ForLoop(n, 0, Model::NGWs - 1,
            (*model.piGWs)(n) += (w * model.dt) *
            ↪  GWsKernels::get(model,n);
    );
}
```

$$\boxed{(\pi_u)'_{ij} = \mathcal{K}_{ij}[u_{ij}, \{\tilde{\phi}\}, a]}$$

```
if (model.fldGWs != nullptr) driftGWs(model);
```

```
template<class Model>
void driftGWs(Model& model) {

    (*model.fldGWs) +=  pow(model.aSI, model.alpha - 3) * (model.dt *
    ↪  (*model.piGWs) );

}
```

$$\boxed{u'_{ij} = a^{\alpha-3}(\pi_u)_{ij}}$$

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: $\mathcal{K}_{ij}[u_{ij}, \{\tilde{\phi}\}, a]$

src/include/CosmoInterface/evolvers/kernels/gwskernels.h

```
        template <class Model, int N>
        static auto get(Model& model, Tag<N> n){

            return (pow(model.aI, 1 + model.alpha) *
                    LatLapl<Model::NDim>( (*model.fldGWs)(n))
                     + pow(model.aI, 1 + model.alpha) * 2 *
                     ↪  pow<2>(model.fStar/Constants::MPl) *
                     ↪  (PITensor::totalTensor(model,n)));
        }
```

$$\boxed{\mathcal{K}_{ij}[u_{ij}, \{\tilde{\phi}\}, a] = a^{1+\alpha}\tilde{\nabla}^2_{\rm L} u_{ij} + 2a^{1+\alpha}\left(\frac{f_*}{m_{\rm Pl}}\right)^2 \tilde{\Pi}_{ij}}$$

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: $\mathcal{K}_{ij}[u_{ij}, \{\tilde{\phi}\}, a]$

src/include/CosmoInterface/evolvers/kernels/gwskernels.h

```
        template <class Model, int N>
        static auto get(Model& model, Tag<N> n){

            return (pow(model.aI, 1 + model.alpha) *
                    LatLapl<Model::NDim>( (*model.fldGWs)(n))
                     + pow(model.aI, 1 + model.alpha) * 2 *
                     ↪  pow<2>(model.fStar/Constants::MPl) *
                     ↪  (PITensor::totalTensor(model,n)));
        }
```

$$\boxed{\mathcal{K}_{ij}[u_{ij}, \{\tilde{\phi}\}, a] = a^{1+\alpha}\tilde{\nabla}^2_{\rm L} u_{ij} + 2a^{1+\alpha}\left(\frac{f_*}{m_{\rm Pl}}\right)^2 \tilde{\Pi}_{ij}}$$

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: $\mathcal{K}_{ij}[u_{ij}, \{\tilde{\phi}\}, a]$

src/include/CosmoInterface/evolvers/kernels/gwskernels.h

```
        template <class Model, int N>
        static auto get(Model& model, Tag<N> n){

            return (pow(model.aI, 1 + model.alpha) *
                    LatLapl<Model::NDim>( (*model.fldGWs)(n))
                    + pow(model.aI, 1 + model.alpha) * 2 * pow<2>(model.fStar/Constants::MPl) * (PITensor::totalTensor(model,n)));
        }
```

$$\boxed{\mathcal{K}_{ij}[u_{ij}, \{\tilde{\phi}\}, a] = a^{1+\alpha}\tilde{\nabla}^2_{\mathrm{L}} u_{ij} + 2a^{1+\alpha}\left(\frac{f_*}{m_{\mathrm{Pl}}}\right)^2 \tilde{\Pi}_{ij}}$$

GWs in CosmoLattice: anisotropic tensor

src/include/CosmoInterface/definitions/PITensor.h

```
    template<class Model>
    static inline auto totalTensor(Model& model, Tag<0>)
    {
        return scalarSinglet(model, 1_c, 1_c) + complexScalar(model, 1_c, 1_c);
    }
    template<class Model>
    static inline auto totalTensor(Model& model, Tag<1>)
    {
        return scalarSinglet(model, 1_c, 2_c) + complexScalar(model, 1_c, 2_c);;
    }
```

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: anisotropic tensor

src/include/CosmoInterface/definitions/PITensor.h

```
    template<class Model>
    static inline auto totalTensor(Model& model, Tag<0>)
    {
        return scalarSinglet(model, 1_c, 1_c) + complexScalar(model, 1_c, 1_c);
    }
    template<class Model>
    static inline auto totalTensor(Model& model, Tag<1>)
    {
        return scalarSinglet(model, 1_c, 2_c) + complexScalar(model, 1_c, 2_c);;
    }
```

```
    template<class Model, int I, int J>
    static inline auto scalarSinglet(Model& model, Tag<I> a, Tag<J> b)
    {
        return Total(i, 0, Model::Ns - 1, forwDiff(model.fldS(i),a) *
        ↪  forwDiff(model.fldS(i),b));
    }
```

$$\boxed{\tilde{\Pi}_{ij} = \sum_a \tilde{\partial}_i \tilde{\phi}_a \tilde{\partial}_j \tilde{\phi}_a}$$

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: $h'_{ij}h'^{*}_{ij}$

src/include/CosmoInterface/definitions/GWsProjector.h

```
template<class Model, class Looper, int I, int J, typename T = double>
inline auto Pr(Model& model, Looper& it, Tag<I> i, Tag<J> j) {

    auto pVec = it.getVec();

    size_t N = GetNGrid::get(model.getOneField());

    auto klattice = MakeVector(l, 1, Model::NDim,
        mType == 1 ? std::sin(2 * Constants::pi<T> * pVec[l-1] / N) :
        mType == 2 ? std::complex<T>(1) - std::complex<T>(std::cos(2.0 *
        ↪  Constants::pi<T> / N * pVec[l-1]),std::sin(2.0 * Constants::pi<T> /
        ↪  N * pVec[l-1])) :
        mType == 3 ? std::complex<T>(1) - std::complex<T>(std::cos(2.0 *
        ↪  Constants::pi<T> / N * pVec[l-1]),std::sin(-2.0 * Constants::pi<T> /
        ↪  N * pVec[l-1])) : std::complex<T>(1.));

    T klatticeSquare = Total(k, 1, Model::NDim,
    ↪  abs((klattice(k))*conj(klattice(k))));

    if(i == j) return (std::complex<T>)(1) - conj(klattice(i)) * (klattice(j)) /
    ↪  klatticeSquare;
    return - conj(klattice(i)) * (klattice(j)) / klatticeSquare;
```

$$\boxed{P_{ij} = \delta_{ij} - \frac{\kappa_{\mathrm{L},i}\kappa^{*}_{\mathrm{L},j}}{\kappa_{\mathrm{L}}^{2}}}$$

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: $h'_{ij}h'^{*}_{ij}$

`src/include/CosmoInterface/definitions/GWsProjector.h`

```
template<class Model, class Looper>
auto projectedGW_complex(Model& model, Looper& it) {

    auto P11 = Pr(model, it, 1_c, 1_c);
    auto P12 = Pr(model, it, 1_c, 2_c);
    auto P13 = Pr(model, it, 1_c, 3_c);
    auto P21 = conj(P12);
    auto P22 = Pr(model, it, 2_c, 2_c);
    auto P23 = Pr(model, it, 2_c, 3_c);
    auto P31 = conj(P13);
    auto P32 = conj(P23);
    auto P33 = Pr(model, it, 3_c, 3_c);

    auto u11 = GetValue::get(model.pi_GWtensor(1_c,1_c).inFourierSpace(), it());
    auto u12 = GetValue::get(model.pi_GWtensor(1_c,2_c).inFourierSpace(), it());
    auto u13 = GetValue::get(model.pi_GWtensor(1_c,3_c).inFourierSpace(), it());
    auto u21 = u12;
    auto u22 = GetValue::get(model.pi_GWtensor(2_c,2_c).inFourierSpace(), it());
    auto u23 = GetValue::get(model.pi_GWtensor(2_c,3_c).inFourierSpace(), it());
    auto u31 = u13;
    auto u32 = u23;
    auto u33 = GetValue::get(model.pi_GWtensor(3_c,3_c).inFourierSpace(), it());
```

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: $h'_{ij}h'^{*}_{ij}$

`src/include/CosmoInterface/definitions/GWsProjector.h`

```
template<class Model, class Looper>
auto projectedGW_complex(Model& model, Looper& it) {

    auto P11 = Pr(model, it, 1_c, 1_c);
    auto P12 = Pr(model, it, 1_c, 2_c);
    auto P13 = Pr(model, it, 1_c, 3_c);
    auto P21 = conj(P12);
    auto P22 = Pr(model, it, 2_c, 2_c);
    auto P23 = Pr(model, it, 2_c, 3_c);
    auto P31 = conj(P13);
    auto P32 = conj(P23);
    auto P33 = Pr(model, it, 3_c, 3_c);

    auto u11 = GetValue::get(model.pi_GWtensor(1_c,1_c).inFourierSpace(), it());
    auto u12 = GetValue::get(model.pi_GWtensor(1_c,2_c).inFourierSpace(), it());
    auto u13 = GetValue::get(model.pi_GWtensor(1_c,3_c).inFourierSpace(), it());
    auto u21 = u12;
    auto u22 = GetValue::get(model.pi_GWtensor(2_c,2_c).inFourierSpace(), it());
    auto u23 = GetValue::get(model.pi_GWtensor(2_c,3_c).inFourierSpace(), it());
    auto u31 = u13;
    auto u32 = u23;
    auto u33 = GetValue::get(model.pi_GWtensor(3_c,3_c).inFourierSpace(), it());
```

$$\boxed{P_{ij} = \delta_{ij} - \frac{\kappa_{\mathrm{L},i}\kappa^{*}_{\mathrm{L},j}}{\kappa^2_{\mathrm{L}}}}$$

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: $h'_{ij} h'^{*}_{ij}$

`src/include/CosmoInterface/definitions/GWsProjector.h`

```
template<class Model, class Looper>
auto projectedGW_complex(Model& model, Looper& it) {

    auto P11 = Pr(model, it, 1_c, 1_c);
    auto P12 = Pr(model, it, 1_c, 2_c);
    auto P13 = Pr(model, it, 1_c, 3_c);
    auto P21 = conj(P12);
    auto P22 = Pr(model, it, 2_c, 2_c);
    auto P23 = Pr(model, it, 2_c, 3_c);
    auto P31 = conj(P13);
    auto P32 = conj(P23);
    auto P33 = Pr(model, it, 3_c, 3_c);

    auto u11 = GetValue::get(model.pi_GWtensor(1_c,1_c).inFourierSpace(), it());
    auto u12 = GetValue::get(model.pi_GWtensor(1_c,2_c).inFourierSpace(), it());
    auto u13 = GetValue::get(model.pi_GWtensor(1_c,3_c).inFourierSpace(), it());
    auto u21 = u12;
    auto u22 = GetValue::get(model.pi_GWtensor(2_c,2_c).inFourierSpace(), it());
    auto u23 = GetValue::get(model.pi_GWtensor(2_c,3_c).inFourierSpace(), it());
    auto u31 = u13;
    auto u32 = u23;
    auto u33 = GetValue::get(model.pi_GWtensor(3_c,3_c).inFourierSpace(), it());
```

$$\boxed{P_{ij} = \delta_{ij} - \frac{\kappa_{\mathrm{L},i}\kappa^{*}_{\mathrm{L},j}}{\kappa^2_{\mathrm{L}}}}$$

$$\boxed{(\pi_u)_{ij}(\kappa, \tilde{\eta}) = a^{3-\alpha} u'_{ij}}$$

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: $h'_{ij}h'^{*}_{ij}$

src/include/CosmoInterface/definitions/GWsProjector.h

```
    auto Pu11 = P11 * u11 + P12 * u21 + P13 * u31;
    auto Pu12 = P11 * u12 + P12 * u22 + P13 * u32;
    auto Pu13 = P11 * u13 + P12 * u23 + P13 * u33;
    auto Pu21 = P21 * u11 + P22 * u12 + P23 * u13;
    auto Pu22 = P21 * u12 + P22 * u22 + P23 * u23;
    auto Pu23 = P21 * u13 + P22 * u23 + P23 * u33;
    auto Pu31 = P31 * u11 + P32 * u21 + P33 * u13;
    auto Pu32 = P31 * u12 + P32 * u22 + P33 * u23;
    auto Pu33 = P31 * u13 + P32 * u23 + P33 * u33;

    auto Pu_s11 = P11 * conj(u11) + P12 * conj(u21) + P13 * conj(u31);
    auto Pu_s12 = P11 * conj(u12) + P12 * conj(u22) + P13 * conj(u32);
    auto Pu_s13 = P11 * conj(u13) + P12 * conj(u23) + P13 * conj(u33);
    auto Pu_s21 = P21 * conj(u11) + P22 * conj(u12) + P23 * conj(u13);
    auto Pu_s22 = P21 * conj(u12) + P22 * conj(u22) + P23 * conj(u23);
    auto Pu_s23 = P21 * conj(u13) + P22 * conj(u23) + P23 * conj(u33);
    auto Pu_s31 = P31 * conj(u11) + P32 * conj(u21) + P33 * conj(u13);
    auto Pu_s32 = P31 * conj(u12) + P32 * conj(u22) + P33 * conj(u23);
    auto Pu_s33 = P31 * conj(u13) + P32 * conj(u23) + P33 * conj(u33);
```

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: $h'_{ij}h'^{*}_{ij}$

src/include/CosmoInterface/definitions/GWsProjector.h

```
    auto Pu11 = P11 * u11 + P12 * u21 + P13 * u31;
    auto Pu12 = P11 * u12 + P12 * u22 + P13 * u32;
    auto Pu13 = P11 * u13 + P12 * u23 + P13 * u33;
    auto Pu21 = P21 * u11 + P22 * u12 + P23 * u13;
    auto Pu22 = P21 * u12 + P22 * u22 + P23 * u23;
    auto Pu23 = P21 * u13 + P22 * u23 + P23 * u33;
    auto Pu31 = P31 * u11 + P32 * u21 + P33 * u13;
    auto Pu32 = P31 * u12 + P32 * u22 + P33 * u23;
    auto Pu33 = P31 * u13 + P32 * u23 + P33 * u33;

    auto Pu_s11 = P11 * conj(u11) + P12 * conj(u21) + P13 * conj(u31);
    auto Pu_s12 = P11 * conj(u12) + P12 * conj(u22) + P13 * conj(u32);
    auto Pu_s13 = P11 * conj(u13) + P12 * conj(u23) + P13 * conj(u33);
    auto Pu_s21 = P21 * conj(u11) + P22 * conj(u12) + P23 * conj(u13);
    auto Pu_s22 = P21 * conj(u12) + P22 * conj(u22) + P23 * conj(u23);
    auto Pu_s23 = P21 * conj(u13) + P22 * conj(u23) + P23 * conj(u33);
    auto Pu_s31 = P31 * conj(u11) + P32 * conj(u21) + P33 * conj(u13);
    auto Pu_s32 = P31 * conj(u12) + P32 * conj(u22) + P33 * conj(u23);
    auto Pu_s33 = P31 * conj(u13) + P32 * conj(u23) + P33 * conj(u33);
```

$\boxed{a^{3-\alpha}Pu'}$

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: $h'_{ij}h'^{*}_{ij}$

src/include/CosmoInterface/definitions/GWsProjector.h

```
    auto Pu11 = P11 * u11 + P12 * u21 + P13 * u31;
    auto Pu12 = P11 * u12 + P12 * u22 + P13 * u32;
    auto Pu13 = P11 * u13 + P12 * u23 + P13 * u33;
    auto Pu21 = P21 * u11 + P22 * u12 + P23 * u13;
    auto Pu22 = P21 * u12 + P22 * u22 + P23 * u23;
    auto Pu23 = P21 * u13 + P22 * u23 + P23 * u33;
    auto Pu31 = P31 * u11 + P32 * u21 + P33 * u13;
    auto Pu32 = P31 * u12 + P32 * u22 + P33 * u23;
    auto Pu33 = P31 * u13 + P32 * u23 + P33 * u33;

    auto Pu_s11 = P11 * conj(u11) + P12 * conj(u21) + P13 * conj(u31);
    auto Pu_s12 = P11 * conj(u12) + P12 * conj(u22) + P13 * conj(u32);
    auto Pu_s13 = P11 * conj(u13) + P12 * conj(u23) + P13 * conj(u33);
    auto Pu_s21 = P21 * conj(u11) + P22 * conj(u12) + P23 * conj(u13);
    auto Pu_s22 = P21 * conj(u12) + P22 * conj(u22) + P23 * conj(u23);
    auto Pu_s23 = P21 * conj(u13) + P22 * conj(u23) + P23 * conj(u33);
    auto Pu_s31 = P31 * conj(u11) + P32 * conj(u21) + P33 * conj(u13);
    auto Pu_s32 = P31 * conj(u12) + P32 * conj(u22) + P33 * conj(u23);
    auto Pu_s33 = P31 * conj(u13) + P32 * conj(u23) + P33 * conj(u33);
```

$\boxed{a^{3-\alpha}Pu'}$

$\boxed{a^{3-\alpha}P(u')^*}$

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: $h'_{ij}h'^{*}_{ij}$

src/include/CosmoInterface/definitions/GWsProjector.h

```
    auto Tr1 = (Pu11 * Pu_s11 + Pu12 * Pu_s21 + Pu13 * Pu_s31) + (Pu21
    ↪  * Pu_s12 + Pu22 * Pu_s22 + Pu23 * Pu_s32) + (Pu31 * Pu_s13 +
    ↪  Pu32 * Pu_s23 + Pu33 * Pu_s33);
    auto Tr2a = (Pu11 + Pu22 + Pu33);
    auto Tr2b = (Pu_s11 + Pu_s22 + Pu_s33);

    return pow(model.aI,2*(model.alpha-3)) * abs(Tr1 - .5 * Tr2a *
    ↪  Tr2b);
}
```

$$\boxed{h'_{ij}h'^{*}_{ij} = \mathrm{Tr}[Pu'P(u')^*] - \frac{1}{2}\mathrm{Tr}[Pu']\mathrm{Tr}[P(u')^*]}$$

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: $h'_{ij}h'^{*}_{ij}$

`src/include/CosmoInterface/definitions/GWsProjector.h`

```
    auto Tr1 = (Pu11 * Pu_s11 + Pu12 * Pu_s21 + Pu13 * Pu_s31) + (Pu21
    ↪  * Pu_s12 + Pu22 * Pu_s22 + Pu23 * Pu_s32) + (Pu31 * Pu_s13 +
    ↪  Pu32 * Pu_s23 + Pu33 * Pu_s33);
    auto Tr2a = (Pu11 + Pu22 + Pu33);
    auto Tr2b = (Pu_s11 + Pu_s22 + Pu_s33);

    return pow(model.aI,2*(model.alpha-3)) * abs(Tr1 - .5 * Tr2a *
    ↪  Tr2b);
}
```

$$\boxed{h'_{ij}h'^{*}_{ij} = \mathrm{Tr}[Pu'P(u')^{*}] - \frac{1}{2}\mathrm{Tr}[Pu']\mathrm{Tr}[P(u')^{*}]}$$

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: $h'_{ij}h'^{*}_{ij}$

`src/include/CosmoInterface/definitions/GWsProjector.h`

```
    auto Tr1 = (Pu11 * Pu_s11 + Pu12 * Pu_s21 + Pu13 * Pu_s31) + (Pu21 * Pu_s12 + Pu22 * Pu_s22 + Pu23 * Pu_s32) + (Pu31 * Pu_s13 + Pu32 * Pu_s23 + Pu33 * Pu_s33);
    auto Tr2a = (Pu11 + Pu22 + Pu33);
    auto Tr2b = (Pu_s11 + Pu_s22 + Pu_s33);

    return pow(model.aI,2*(model.alpha-3)) * abs(Tr1 - .5 * Tr2a * Tr2b);
}
```

$$\boxed{h'_{ij}h'^{*}_{ij} = \mathrm{Tr}[Pu'P(u')^*] - \frac{1}{2}\mathrm{Tr}[Pu']\mathrm{Tr}[P(u')^*]}$$

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: $h'_{ij}h'^{*}_{ij}$

src/include/CosmoInterface/definitions/GWsProjector.h

```
    auto Tr1 = (Pu11 * Pu_s11 + Pu12 * Pu_s21 + Pu13 * Pu_s31) + (Pu21 * Pu_s12 + Pu22 * Pu_s22 + Pu23 * Pu_s32) + (Pu31 * Pu_s13 + Pu32 * Pu_s23 + Pu33 * Pu_s33);
    auto Tr2a = (Pu11 + Pu22 + Pu33);
    auto Tr2b = (Pu_s11 + Pu_s22 + Pu_s33);

    return pow(model.aI,2*(model.alpha-3)) * abs(Tr1 - .5 * Tr2a * Tr2b);
}
```

$$\boxed{h'_{ij}h'^{*}_{ij} = \mathrm{Tr}[Pu'P(u')^{*}] - \frac{1}{2}\mathrm{Tr}[Pu']\mathrm{Tr}[P(u')^{*}]}$$

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: $h'_{ij}h'^{*}_{ij}$

`src/include/CosmoInterface/definitions/GWsProjector.h`

```
    auto Tr1 = (Pu11 * Pu_s11 + Pu12 * Pu_s21 + Pu13 * Pu_s31) + (Pu21 * Pu_s12 + Pu22 * Pu_s22 + Pu23 * Pu_s32) + (Pu31 * Pu_s13 + Pu32 * Pu_s23 + Pu33 * Pu_s33);
    auto Tr2a = (Pu11 + Pu22 + Pu33);
    auto Tr2b = (Pu_s11 + Pu_s22 + Pu_s33);

    return pow(model.aI,2*(model.alpha-3)) * abs(Tr1 - .5 * Tr2a * Tr2b);
}
```

$$\boxed{h'_{ij}h'^{*}_{ij} = \mathrm{Tr}[Pu'P(u')^*] - \frac{1}{2}\mathrm{Tr}[Pu']\mathrm{Tr}[P(u')^*]}$$

Similar for the real projector

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: Ω_{GW}

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: $\Omega_{\rm GW}$

`src/include/CosmoInterface/measurements/gwspowerspectrum.h`

see Technical note I: Power Spectrum

```
    if (PSVersion != 3){
        auto fk2 = projectRadiallyGW(model, PSVersion == 1, PRJType, PSType,
        ↪  PSVersion).measure(model, nbins, kMaxBins);
        if (PSType == 2){
            return Function(ntilde, pow<3>(kIR * ntilde * dx ) / N3
            ↪  /(8*pow<2>(Constants::pi<T>)*pow(model.aI,2*model.alpha))
            ↪  *pow<2>(Constants::reducedMPlanck<T> / model.fStar)/Energies::rho(model)) *
            ↪  fk2;
        }
```

$$\boxed{\Omega_{\rm GW} = \frac{1}{\tilde{\rho}_{\rm tot}} \frac{\kappa^3}{8\pi^2 a^{2\alpha}} \left(\frac{\delta\tilde{x}}{N}\right)^3 \left(\frac{m_{\rm Pl}}{f_*}\right)^2 \langle h'_{ij} h'^{*}_{ij} \rangle_{R(\kappa)}}$$

GWs in $\mathcal{C}$osmo$\mathcal{L}$attice: $\Omega_{\rm GW}$

`src/include/CosmoInterface/measurements/gwspowerspectrum.h`

see Technical note I: Power Spectrum

```
    if (PSVersion != 3){
        auto fk2 = projectRadiallyGW(model, PSVersion == 1, PRJType, PSType,
        ↪  PSVersion).measure(model, nbins, kMaxBins);
        if (PSType == 2){
            return Function(ntilde, pow<3>(kIR * ntilde * dx ) / N3
            ↪  /(8*pow<2>(Constants::pi<T>)*pow(model.aI,2*model.alpha))
            ↪  *pow<2>(Constants::reducedMPlanck<T> / model.fStar)/Energies::rho(model)) *
            ↪  fk2;
        }
```

$$\boxed{\Omega_{\rm GW} = \frac{1}{\tilde{\rho}_{\rm tot}} \frac{\kappa^3}{8\pi^2 a^{2\alpha}} \left(\frac{\delta\tilde{x}}{N}\right)^3 \left(\frac{m_{\rm Pl}}{f_*}\right)^2 \langle h'_{ij} h'^{*}_{ij} \rangle_{R(\kappa)}}$$

```
        else if (PSType == 1){
            fk2.sumInsteadOfAverage();
            return Function(ntilde, kIR * ntilde * dx  / pow<5>(N) /
            ↪  (8*(Constants::pi<T>)*pow(model.aI,2*model.alpha))
            ↪  *pow<2>(Constants::reducedMPlanck<T> / model.fStar)/Energies::rho(model)) *
            ↪  fk2;
        }
```

$$\boxed{\Omega_{\rm GW} = \frac{1}{\tilde{\rho}_{\rm tot}} \frac{\kappa}{8\pi^2 a^{2\alpha}} \left(\frac{\delta\tilde{x}}{N^5}\right) \left(\frac{m_{\rm Pl}}{f_*}\right)^2 \#_\kappa \langle h'_{ij} h'^{*}_{ij} \rangle_{R(\kappa)}}$$