Gravitational Wave module

Jorge Baeza-Ballesteros
Nicolas Loayza Romero

Lecture is based on
Technical Note II: Gravitational Waves

Available on: https://cosmolattice.net/technicalnotes/

CosmoLattice

A modern code for lattice simulations of scalar and gauge field dynamics in an expanding universe

What is CosmoLattice?

CosmoLattice is a modern package for **lattice simulations of field dynamics in an expanding universe**. We have developed CosmoLattice to provide a new up-to-date, relevant numerical tool for the scientific community working in the **physics of the early universe**.

CosmoLattice can simulate the dynamics of i) interacting scalar field theories, ii) Abelian U(1) gauge theories, and iii) non-Abelian SU(2) gauge theories, either in flat spacetime or an expanding FLRW background, including the case of self-consistent expansion sourced by the fields themselves. CosmoLattice is ready to simulate the dynamics of field theories described by an action and a background metric of the type:

$$S = -\int d^4x \left\{ \frac{1}{2}\partial_\mu\phi\partial^\mu\phi + (D^A_\mu\varphi)^*(D^\mu_A\varphi) + (D_\mu\Phi)^\dagger(D^\mu\Phi) + \frac{1}{4}F_{\mu\nu}F^{\mu\nu} + \frac{1}{2}\mathrm{Tr}\{G_{\mu\nu}G^{\mu\nu}\} + V(\phi,|\varphi|,|\Phi|) \right\}$$

DOCUMENTATION

IN A NUTSHELL

USER MANUAL

MONOGRAPHIC REVIEW

TECHNICAL NOTES

Technical notes

Here are some technical notes that expand some of the contents of the user manual, and/or explain new functionalities incorporated to the code in successive releases.

- **Technical Note I: Power spectra**
  Daniel G. Figueroa and Adrien Florio. Released on 06.05.2022.

- **Technical Note II: Gravitational Waves**
  Jorge Baeza-Ballesteros, Daniel G. Figueroa, Adrien Florio, Nicolas Loayza.
  Released on 06.05.2022.

Part I: Theoretical basis of Gravitational Waves

Gravitational Wave equation of motion

FLRW background + tensor perturbations

$$ds^2 = -dt^2 + a(t)^2(\delta_{ij} + h_{ij})dx_i dx_j \qquad h_{ii} = \partial_i h_{ij} = 0$$

$$\ddot{h}_{ij} + 3H\dot{h}_{ij} - \frac{\nabla^2}{a^2}h_{ij} = \frac{2}{m_p^2 a^2}\{\Pi_{ij}\}^{TT}$$

Gravitational Wave equation of motion

FLRW background + tensor perturbations

$$ds^2 = -dt^2 + a(t)^2(\delta_{ij} + h_{ij})dx_i dx_j \qquad h_{ii} = \partial_i h_{ij} = 0$$

$$\ddot{h}_{ij} + 3H\dot{h}_{ij} - \frac{\nabla^2}{a^2}h_{ij} = \frac{2}{m_p^2 a^2}\{\Pi_{ij}\}^{TT}$$

Latin indices $\{i, j, \ldots\}$ spatial components

Summation over repeated indices

Gravitational Wave equation of motion

$$\ddot{h}_{ij} + 3H\dot{h}_{ij} - \frac{\nabla^2}{a^2}h_{ij} = \frac{2}{m_p^2 a^2}\{\Pi_{ij}\}^{TT}$$

Gravitational Wave equation of motion

$$\ddot{h}_{ij} + 3H\dot{h}_{ij} - \frac{\nabla^2}{a^2} h_{ij} = \frac{2}{m_p^2 a^2} \{\Pi_{ij}\}^{TT}$$

TT part of Anisotropic stress tensor

Gravitational Wave equation of motion

$$\ddot{h}_{ij} + 3H\dot{h}_{ij} - \frac{\nabla^2}{a^2}h_{ij} = \frac{2}{m_p^2 a^2}\{\Pi_{ij}\}^{TT}$$

TT part of Anisotropic stress tensor

$$\partial_i \Pi_{ij}^{TT} = \Pi_{ii}^{TT} = 0 \text{ hold } \forall \mathbf{x}, \forall \mathbf{t}$$

Gravitational Wave equation of motion

$$\ddot{h}_{ij} + 3H\dot{h}_{ij} - \frac{\nabla^2}{a^2} h_{ij} = \frac{2}{m_p^2 a^2} \{\Pi_{ij}\}^{TT}$$

TT part of Anisotropic stress tensor

$$\partial_i \Pi_{ij}^{TT} = \Pi_{ii}^{TT} = 0 \text{ hold } \forall \mathbf{x}, \forall \mathbf{t}$$

To obtain TT part of a tensor in real space is a **non-local operation!**

TT projection in Fourier Space

$$f(\mathbf{x}, t) = \int \frac{d^3\mathbf{k}}{(2\pi)^3} \; e^{i\mathbf{x}\cdot\mathbf{k}} \; \tilde{f}(\mathbf{k}, t)$$

Continuum Fourier Transform

TT projection in Fourier Space

$$f(\mathbf{x},t) = \int \frac{d^3\mathbf{k}}{(2\pi)^3}\; e^{i\mathbf{x}\cdot\mathbf{k}}\; \tilde{f}(\mathbf{k},t)$$

Continuum Fourier Transform

$$\Pi_{ij}^{TT}(\mathbf{k}) = \Lambda_{ijkl}(\hat{\mathbf{k}})\Pi_{kl}(\mathbf{k},t)$$

$$\Lambda_{ij,lm}(\hat{\mathbf{k}}) \equiv P_{il}(\hat{\mathbf{k}})P_{jm}(\hat{\mathbf{k}}) - \frac{1}{2}P_{ij}(\hat{\mathbf{k}})P_{lm}(\hat{\mathbf{k}}) \quad \text{with} \quad P_{ij} = \delta_{ij} - \hat{\mathbf{k}}_i\hat{\mathbf{k}}_j\,,\ \hat{\mathbf{k}}_i = \mathbf{k}_i/k$$

TT projection in Fourier Space

$$f(\mathbf{x},t) = \int \frac{d^3\mathbf{k}}{(2\pi)^3}\; e^{i\mathbf{x}\cdot\mathbf{k}}\; \tilde{f}(\mathbf{k},t)$$

Continuum Fourier Transform

$$\Pi_{ij}^{TT}(\mathbf{k}) = \Lambda_{ij,lm}(\hat{\mathbf{k}})\Pi_{lm}(\mathbf{k},t)$$

check $\Pi_{ij}^{TT}(\mathbf{k})$ satisfies $k_i\Pi_{ij}^{TT} = \Pi_{ii}^{TT} = 0$

$$\Lambda_{ij,lm}(\mathbf{k}) \equiv P_{il}(\mathbf{k})P_{jm}(\mathbf{k}) - \frac{}{2}P_{ij}(\mathbf{k})P_{lm}(\mathbf{k}) \quad \text{with} \quad P_{ij} = \delta_{ij} - \mathbf{k}_i\mathbf{k}_j\,,\ \mathbf{k}_i = \mathbf{k}_i/k$$

TT projection in Fourier Space

$$f(\mathbf{x},t) = \int \frac{d^3\mathbf{k}}{(2\pi)^3}\; e^{i\mathbf{x}\cdot\mathbf{k}}\; \tilde{f}(\mathbf{k},t)$$

Continuum Fourier Transform

$$\Pi_{ij}^{TT}(\mathbf{k}) = \Lambda_{ijkl}(\hat{\mathbf{k}})\Pi_{kl}(\mathbf{k},t)$$

$$\Lambda_{ij,lm}(\hat{\mathbf{k}}) \equiv P_{il}(\hat{\mathbf{k}})P_{jm}(\hat{\mathbf{k}}) - \frac{1}{2}P_{ij}(\hat{\mathbf{k}})P_{lm}(\hat{\mathbf{k}}) \quad \text{with} \quad P_{ij} = \delta_{ij} - \hat{\mathbf{k}}_i\hat{\mathbf{k}}_j\,,\ \hat{\mathbf{k}}_i = \mathbf{k}_i/k$$

$$\Pi_{ij}^{TT}(\mathbf{x},t) = \int \frac{d^3\mathbf{k}}{(2\pi)^3}\, e^{i\mathbf{x}\cdot\mathbf{k}}\Lambda_{ijkl}(\hat{\mathbf{k}}) \int d^3\mathbf{x}\, e^{-i\mathbf{k}\cdot\mathbf{x}}\, \Pi_{kl}(\mathbf{x},t)$$

TT projection in Fourier Space

$$f(\mathbf{x},t) = \int \frac{d^3\mathbf{k}}{(2\pi)^3}\; e^{i\mathbf{x}\cdot\mathbf{k}}\; \tilde{f}(\mathbf{k},t)$$

Continuum Fourier Transform

$$\Pi_{ij}^{TT}(\mathbf{k}) = \Lambda_{ijkl}(\hat{\mathbf{k}})\Pi_{kl}(\mathbf{k},t)$$

$$\Lambda_{ij,lm}(\hat{\mathbf{k}}) \equiv P_{il}(\hat{\mathbf{k}})P_{jm}(\hat{\mathbf{k}}) - \frac{1}{} \quad \text{with} \quad P_{ij} = \delta_{ij} - \hat{\mathbf{k}}_i\hat{\mathbf{k}}_j\,,\ \hat{\mathbf{k}}_i = \mathbf{k}_i/k$$

Problem!

$$\Pi_{ij}^{TT}(\mathbf{x},t) = \int \frac{d^3\mathbf{k}}{(2\pi)^3}\, e^{i\mathbf{x}\cdot\mathbf{k}}\Lambda_{ijkl}(\hat{\mathbf{k}}) \int d^3\mathbf{x}\, e^{-i\mathbf{k}\cdot\mathbf{x}}\, \Pi_{kl}(\mathbf{x},t)$$

Anisotropic stress tensor

$$\Pi_{\mu\nu} \equiv T_{\mu\nu} - (T_{\mu\nu})_{perfect\,fluid}$$

Spatial Component $\Pi_{ij} \equiv T_{ij} - p g_{ij}$

p = homogeneous background pressure

$$g_{ij} = a^2(t)(\delta_{ij} + h_{ij})$$

Anisotropic stress tensor

$$\Pi_{\mu\nu} \equiv T_{\mu\nu} - (T_{\mu\nu})_{perfect\,fluid}$$

Spatial Component $\quad \Pi_{ij} \equiv T_{ij} - p g_{ij}$

p = homogeneous background pressure

$$g_{ij} = a^2(t)(\delta_{ij} + h_{ij})$$

Example

Scalar fields $\quad \Pi_{ij}^{TT} = \left\{ \sum_a \partial_i \phi_a \partial_j \phi_a \right\}^{TT}$

Energy density of Stochastic Gravitational Wave Background (SGWB)

$$\rho_{\mathrm{GW}}(t) = \frac{1}{32\pi G}\langle \dot{h}_{ij}(\mathbf{x}, t)\dot{h}_{ij}(\mathbf{x}, t)\rangle_V$$

$\langle \ldots \rangle_V$ Volume average to encompass all relevant wavelengths

Energy density of Stochastic Gravitational Wave Background (SGWB)

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

$$\approx \frac{1}{32\pi G V}\int_V \frac{d^3\mathbf{k}}{(2\pi)^3}\dot{h}_{ij}(\mathbf{k},t)\dot{h}^*_{ij}(\mathbf{k},t)$$

Fourier transform of $h_{ij}(\mathbf{x},t)$,

Approximation valid for

$kV^{1/3} \gg 1$

Energy density of Stochastic Gravitational Wave Background (SGWB)

$$\rho_{\mathrm{GW}}(t) = \frac{1}{32\pi G}\langle \dot{h}_{ij}(\mathbf{x},t)\dot{h}_{ij}(\mathbf{x},t)\rangle_V$$

$$\approx \frac{1}{32\pi GV}\int_V \frac{d^3\mathbf{k}}{(2\pi)^3}\dot{h}_{ij}(\mathbf{k},t)\dot{h}^*_{ij}(\mathbf{k},t)$$

$$\rho_{\mathrm{GW}}(t) = \int \frac{d\rho_{\mathrm{GW}}(k,t)}{d\log k}\frac{dk}{k}$$

Energy density of Stochastic Gravitational Wave Background (SGWB)

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

$$\approx \frac{1}{32\pi GV}\int_V \frac{d^3\mathbf{k}}{(2\pi)^3}\dot{h}_{ij}(\mathbf{k},t)\dot{h}^*_{ij}(\mathbf{k},t)$$

Energy density per logarithmic interval

$$\frac{d\rho_{\rm GW}}{d\log k} = \frac{k^3}{(4\pi)^3 GV}\int \frac{d\Omega_k}{4\pi}\dot{h}_{ij}(\mathbf{k},t)\dot{h}^*_{ij}(\mathbf{k},t)$$

Energy density of Stochastic Gravitational Wave Background (SGWB)

$$\rho_{\mathrm{GW}}(t) = \frac{1}{32\pi G}\langle \dot{h}_{ij}(\mathbf{x}, t)\dot{h}^{*}_{ij}(\mathbf{x}, t)\rangle$$

Energy density of Stochastic Gravitational Wave Background (SGWB)

$$\rho_{\rm GW}(t) = \frac{1}{32\pi G}\langle \dot{h}_{ij}(\mathbf{x},t)\dot{h}^{*}_{ij}(\mathbf{x},t)\rangle$$

For stochastic sources use ensemble average $\langle \ldots \rangle$

Energy density of Stochastic Gravitational Wave Background (SGWB)

$$\rho_{\rm GW}(t) = \frac{1}{32\pi G}\langle \dot{h}_{ij}(\mathbf{x},t)\dot{h}^{*}_{ij}(\mathbf{x},t)\rangle$$

$$\langle \dot{h}_{ij}(\mathbf{k},t)\dot{h}^{*}_{ij}(\mathbf{k}',t)\rangle = (2\pi)^3 P_{\dot{h}}(k,t)\delta^{(3)}(\mathbf{k}-\mathbf{k}')$$

$$\rho_{\rm GW}(t) \equiv \frac{1}{(4\pi)^3 G}\int \frac{dk}{k} k^3 P_{\dot{h}}(k,t)$$

$P_{\dot{h}}$ power spectrum of $\dot{h}$

Energy density of Stochastic Gravitational Wave Background (SGWB)

$$\rho_{\rm GW}(t) = \frac{1}{32\pi G}\langle \dot{h}_{ij}(\mathbf{x},t)\dot{h}^*_{ij}(\mathbf{x},t)\rangle$$

$$\langle \dot{h}_{ij}(\mathbf{k},t)\dot{h}^*_{ij}(\mathbf{k}',t)\rangle = (2\pi)^3 P_{\dot{h}}(k,t)\delta^{(3)}(\mathbf{k}-\mathbf{k}')$$

$$\rho_{\rm GW}(t) \equiv \frac{1}{(4\pi)^3 G}\int \frac{dk}{k} k^3 P_{\dot{h}}(k,t)$$

Energy density per logarithmic interval

$$\frac{d\rho_{\rm GW}}{d\log k} = \frac{k^3}{(4\pi)^3 G} P_{\dot{h}}(k,t)$$

Energy density of Stochastic Gravitational Wave Background (SGWB)

Energy density per logarithmic interval

$$\frac{d\rho_{\rm GW}}{d\log k} = \frac{k^3}{(4\pi)^3 GV}\int \frac{d\Omega_k}{4\pi}\dot{h}_{ij}(\mathbf{k},t)\dot{h}^*_{ij}(\mathbf{k},t)$$

Critical energy density

$$\rho_{\rm c} \equiv 3H^2/8\pi G$$

GW energy density per critical energy density

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

Gravitational Waves equation of motion

$$\ddot{h}_{ij} + 3H\dot{h}_{ij} + \frac{\nabla^2}{a^2} h_{ij} = \frac{2}{m_p^2 a^2} \underbrace{\{\Pi_{ij}\}^{TT}}$$

Transverse-Traceless Tensor

$$\partial_i \Pi_{ij}^{TT} \quad \text{Problem!} \quad \mathbf{x}, \forall \mathbf{t}$$

To obtain TT part of a tensor in real space is a **non-local operation!**

$$\Pi_{ij}^{TT}(\mathbf{x}, t) = \int \frac{d^3\mathbf{k}}{(2\pi)^3} e^{i\mathbf{x}\cdot\mathbf{k}} \Lambda_{ijkl}(\hat{\mathbf{k}}) \int d^3\mathbf{x}\, e^{-i\mathbf{k}\cdot\mathbf{x}}\, \Pi_{kl}(\mathbf{x}, t)$$

Evolution of GW modes

non-local operations are computationally expensive!

Every Time-step!

$$\Pi_{ij}^{TT}(\mathbf{x},t) = \int \frac{d^3\mathbf{k}}{(2\pi)^3}\, e^{i\mathbf{x}\cdot\mathbf{k}} \Lambda_{ijkl}(\hat{\mathbf{k}}) \int d^3\mathbf{x}\, e^{-i\mathbf{k}\cdot\mathbf{x}}\, \Pi_{kl}(\mathbf{x},t)$$

Evolution of GWs modes

non-local operations are computationally expensive!

Solution: we define a set of unphysical tensor modes u's

1) Evolve equation of motion of u's

$$\ddot{u}_{ij} + 3H\dot{u}_{ij} - \frac{\nabla^2}{a^2}u_{ij} = \frac{2}{m_p^2 a^2}\Pi_{ij}$$

Garcia-Bellido, Figueroa & Sastre '07

Evolution of GWs modes

non-local operations are computationally expensive!

Solution: we define a set of unphysical tensor modes u's

1) Evolve equation of motion of u's

$$\ddot{u}_{ij} + 3H\dot{u}_{ij} - \frac{\nabla^2}{a^2}u_{ij} = \frac{2}{m_p^2 a^2}\Pi_{ij}$$

2) When needed, (compute power spectrum energy density) we apply transformation

$$h_{ij}(k,t) = \Lambda_{ij,kl}(k)u_{kl}(k,t)$$

Garcia-Bellido, Figueroa & Sastre '07

Why it works?

$$h_{ij}(k,t) = \Lambda_{ij,kl}(k) u_{kl}(k,t)$$

$$\ddot{h}_{ij} + 3H\dot{h}_{ij} - \frac{\nabla^2}{a^2} h_{ij} = \frac{2}{m_p^2 a^2} \{\Pi_{ij}\}^{TT}$$

Linear in Fourier Space
=> commute

Why it works?

$$h_{ij}(k,t) = \Lambda_{ij,kl}(k) u_{kl}(k,t)$$

$$\left(\frac{d^2}{dt^2} + 3H\frac{d}{dt} + \frac{k^2}{a^2} \right) h_{ij}(k,t) = \frac{2}{m_p^2 a^2} \{\Pi_{ij}\}^{TT}(k,t)$$

Why it works?

$$h_{ij}(k,t) = \Lambda_{ij,kl}(k) u_{kl}(k,t)$$

$$\left(\frac{d^2}{dt^2} + 3H\frac{d}{dt} + \frac{k^2}{a^2} \right) \Lambda_{ijkl} u_{kl}\,(k,t) = \frac{2}{m_p^2 a^2} \Lambda_{ijkl} \Pi_{kl}(k,t)$$

Why it works?

$$\ddot{u}_{ij} + 3H\dot{u}_{ij} - \frac{\nabla^2}{a^2}u_{ij} = \frac{2}{m_p^2 a^2}\Pi_{ij}$$

Every Time-step!

$$h_{ij}(k,t) = \Lambda_{ij,kl}(k)u_{kl}(k,t)$$

Few Times!

Energy density of Stochastic Gravitational Wave Background (SGWB)

Energy density per logarithmic interval

$$\frac{d\rho_{\rm GW}}{d\log k} = \frac{k^3}{(4\pi)^3 GV}\int \frac{d\Omega_k}{4\pi}\dot{h}_{ij}(\mathbf{k},t)\dot{h}^*_{ij}(\mathbf{k},t)$$

Energy density of Stochastic Gravitational Wave Background (SGWB)

Energy density per logarithmic interval

$$\frac{d\rho_{\mathrm{GW}}}{d\log k} = \frac{k^3}{(4\pi)^3 GV}\int \frac{d\Omega_k}{4\pi}\, \dot{h}_{ij}(\mathbf{k},t)\dot{h}^*_{ij}(\mathbf{k},t)$$

$$\begin{aligned}\dot{h}_{ij}(\mathbf{k},t)\dot{h}^*_{ij}(\mathbf{k},t) &= (\Lambda_{ijkl}(\mathbf{k})u_{kl}(\mathbf{k},t))(\Lambda_{ijmn}(\mathbf{k})u^*_{mn}(\mathbf{k},t)) \\ &= \Lambda_{klmn}(\mathbf{k})u_{kl}(\mathbf{k},t)u^*_{mn}(\mathbf{k},t)\end{aligned}$$

Energy density of Stochastic Gravitational Wave Background (SGWB)

Energy density per logarithmic interval

$$\frac{d\rho_{\rm GW}}{d\log k} = \frac{k^3}{(4\pi)^3 GV}\int \frac{d\Omega_k}{4\pi}\, \dot{h}_{ij}(\mathbf{k},t)\dot{h}^*_{ij}(\mathbf{k},t)$$

$$\dot{h}_{ij}(\mathbf{k},t)\dot{h}^*_{ij}(\mathbf{k},t) = (\Lambda_{ijkl}(\mathbf{k})u_{kl}(\mathbf{k},t))(\Lambda_{ijmn}(\mathbf{k})u^*_{mn}(\mathbf{k},t))$$

$$= \Lambda_{klmn}(\mathbf{k})u_{kl}(\mathbf{k},t)u^*_{mn}(\mathbf{k},t)$$

$$\Lambda_{klmn}(\mathbf{k}) = \Lambda_{klpq}(\mathbf{k})\Lambda_{pqmn}(\mathbf{k})$$

Energy density of Stochastic Gravitational Wave Background (SGWB)

$$\dot{h}_{ij}(\mathbf{k},t)\dot{h}^{*}_{ij}(\mathbf{k},t) = \mathrm{Tr}(\mathtt{P}\,\dot{\mathtt{u}}\,\mathtt{P}\,\dot{\mathtt{u}}^{*}) - \frac{1}{2}\mathrm{Tr}(\mathtt{P}\,\dot{\mathtt{u}})\mathrm{Tr}(\mathtt{P}\,\dot{\mathtt{u}}^{*})$$

$$(\dot{\mathtt{u}})_{ij} = \dot{u}_{ij} \qquad (\mathtt{P})_{ij} = P_{ij}$$

$$\Lambda_{ij,lm}(\hat{\mathbf{k}}) \equiv P_{il}(\hat{\mathbf{k}})P_{jm}(\hat{\mathbf{k}}) - \frac{1}{2}P_{ij}(\hat{\mathbf{k}})P_{lm}(\hat{\mathbf{k}}) \quad \text{with} \quad P_{ij} = \delta_{ij} - \hat{\mathbf{k}}_i\hat{\mathbf{k}}_j\,,\ \hat{\mathbf{k}}_i = \mathbf{k}_i/k$$

Energy density of Stochastic Gravitational Wave Background (SGWB)

$$\dot{h}_{ij}(\mathbf{k},t)\dot{h}^{*}_{ij}(\mathbf{k},t) = \mathrm{Tr}(\mathrm{P}\,\dot{\mathrm{u}}\,\mathrm{P}\,\dot{\mathrm{u}}^{*}) - \frac{1}{2}\mathrm{Tr}(\mathrm{P}\,\dot{\mathrm{u}})\mathrm{Tr}(\mathrm{P}\,\dot{\mathrm{u}}^{*})$$

$$v_{ij} \equiv P_{ik}\dot{u}_{kj} \qquad \tilde{v}_{ij} \equiv P_{ik}\dot{u}^{*}_{kj}$$

$$\mathrm{Tr}(\mathrm{P}\,\dot{\mathrm{u}}\,\mathrm{P}\,\dot{\mathrm{u}}^{*}) = v_{11}\tilde{v}_{11} + v_{22}\tilde{v}_{22} + v_{33}\tilde{v}_{33} + v_{12}\tilde{v}_{21} + v_{21}\tilde{v}_{12} + v_{13}\tilde{v}_{31} + v_{31}\tilde{v}_{13} + v_{23}\tilde{v}_{32} + v_{32}\tilde{v}_{23}$$

$$\mathrm{Tr}(\mathrm{P}\,\dot{\mathrm{u}}) = v_{11} + v_{22} + v_{33}$$

$$\mathrm{Tr}(\mathrm{P}\,\dot{\mathrm{u}}^{*}) = \tilde{v}_{11} + \tilde{v}_{22} + \tilde{v}_{33}$$

Energy density of Stochastic Gravitational Wave Background (SGWB)

$$\dot{h}_{ij}(\mathbf{k},t)\dot{h}^{*}_{ij}(\mathbf{k},t) = \mathrm{Tr}(\mathrm{P}\,\dot{\mathrm{u}}\,\mathrm{P}\,\dot{\mathrm{u}}^{*}) - \frac{1}{2}\mathrm{Tr}(\mathrm{P}\,\dot{\mathrm{u}})\mathrm{Tr}(\mathrm{P}\,\dot{\mathrm{u}}^{*})$$

$$v_{ij} \equiv P_{ik}\dot{u}_{kj} \qquad \tilde{v}_{ij} \equiv P_{ik}\dot{u}^{*}_{kj}$$

$$\mathrm{Tr}(\mathrm{P}\,\dot{\mathrm{u}}\,\mathrm{P}\,\dot{\mathrm{u}}^{*}) = v_{11}\tilde{v}_{11} + v_{22}\tilde{v}_{22} + v_{33}\tilde{v}_{33} + v_{12}\tilde{v}_{21} + v_{21}\tilde{v}_{12} + v_{13}\tilde{v}_{31} + v_{31}\tilde{v}_{13} + v_{23}\tilde{v}_{32} + v_{32}\tilde{v}_{23}$$

$$\mathrm{Tr}(\mathrm{P}\,\dot{\mathrm{u}}) = v_{11} + v_{22} + v_{33}$$

$$\mathrm{Tr}(\mathrm{P}\,\dot{\mathrm{u}}^{*}) = \tilde{v}_{11} + \tilde{v}_{22} + \tilde{v}_{33}$$

Notice in Continuum $\tilde{v}_{ij} = v^{*}_{ij}$

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

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

Recap of lattice discretization (II)

Recap of lattice discretization (II)

We define a lattice momentum

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

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

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

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

GWs in the lattice

GWs in the lattice

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

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

GWs in the lattice

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

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

Physical fields projected **only** when measuring

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

GWs in the lattice

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

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

Physical fields projected **only** when measuring

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

Need to choose a particular lattice derivative

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

Ensures $h_{ii} = 0$ and $\nabla_{\mathrm{L},i} h_{ij} = 0$ [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

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

$$= \frac{1}{32\pi G N^6}\sum_{\tilde{\boldsymbol{n}}} \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t)$$

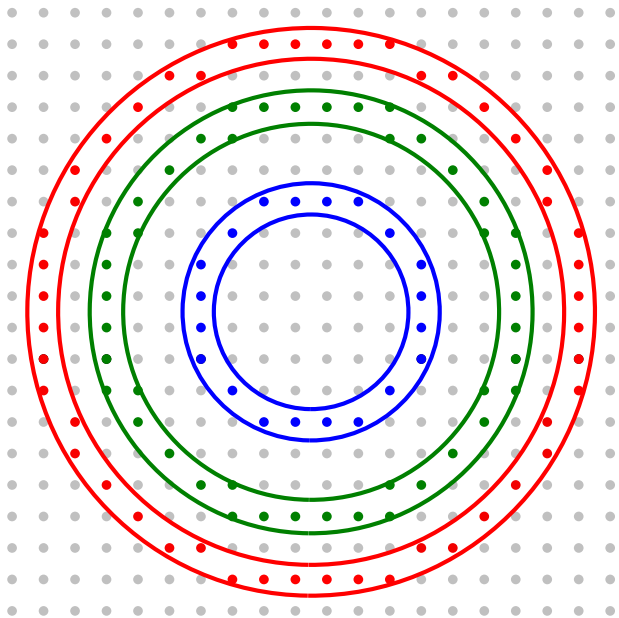
$\downarrow$

Fourier transform
Parseval's theorem

GW energy density power spectrum

We define the GW energy density as a volume average

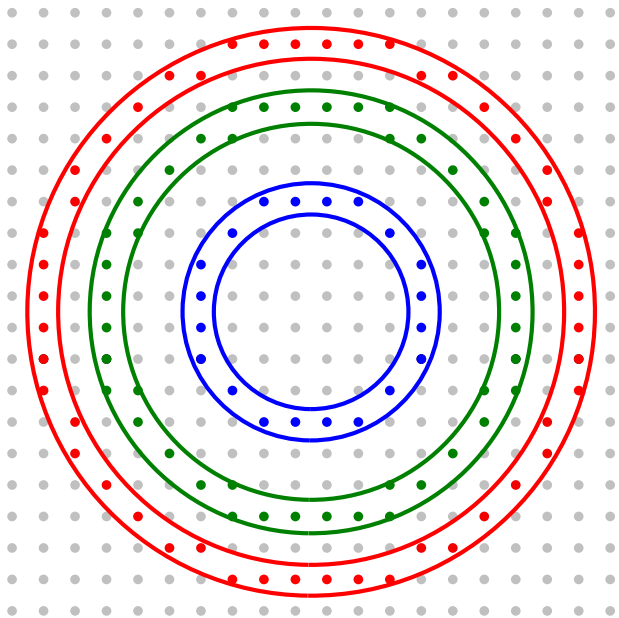
$$\begin{aligned}\rho_{\rm GW}(t) &= \frac{1}{32\pi G}\langle \dot{h}_{ij}\dot{h}_{ij}\rangle_V = \frac{1}{32\pi G N^3}\sum_{\boldsymbol{n}} \dot{h}_{ij}(\boldsymbol{n},t)\dot{h}_{ij}(\boldsymbol{n},t)\\ &= \frac{1}{32\pi G N^6}\sum_{\tilde{\boldsymbol{n}}} \dot{h}_{ij}(\tilde{\boldsymbol{n}},t)\dot{h}^*_{ij}(\tilde{\boldsymbol{n}},t)\\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



GW energy density power spectrum

We define the GW energy density as a volume average

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

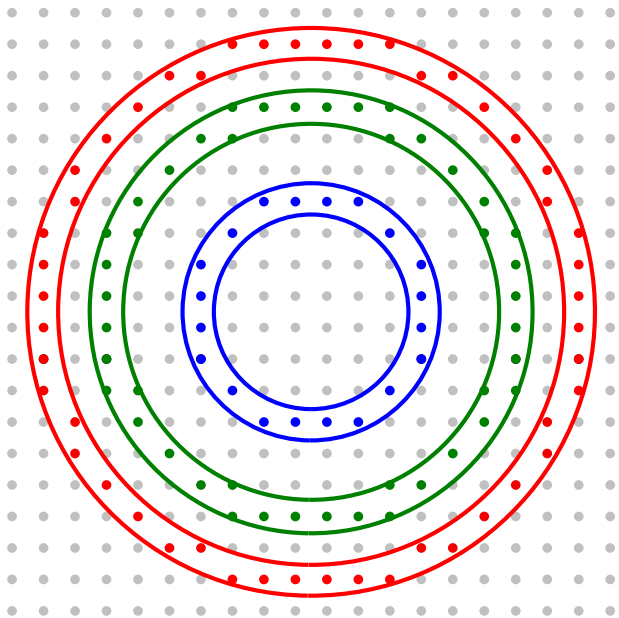


Multiplicity of shells
$\approx 4\pi\tilde{n}^2$

GW energy density power spectrum

We define the GW energy density as a volume average

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



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

GW energy density power spectrum

We define the GW energy density as a volume average

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

GW energy density power spectrum

We define the GW energy density as a volume average

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

GW energy density power spectrum

We define the GW energy density as a volume average

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

Using the TT projector

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

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

GW energy density power spectrum

We define the GW energy density as a volume average

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

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

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

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

```
13 #Times
14 tOutputFreq = 1
15 tOutputInfreq = 1
16 tMax = 300
17
18 #Spectra options
19 PS_type = 1
20 PS_version = 1
21
22 #GWs
23 GWprojectorType = 2
24 withGWs=true
25
26 #IC
27 kCutOff = 4
28 initial_amplitudes = 5.6964e18 0  # homogeneous amplitudes in GeV
29 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

```
13  #Times
14  tOutputFreq = 1
15  tOutputInfreq = 1
16  tMax = 300
17
18  #Spectra options
19  PS_type = 1
20  PS_version = 1
21
22  #GWs
23  GWprojectorType = 2
24  withGWs=true
25
26  #IC
27  kCutOff = 4
28  initial_amplitudes = 5.6964e18 0  # homogeneous amplitudes in GeV
29  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`

```
13  #Times
14  tOutputFreq = 1
15  tOutputInfreq = 1
16  tMax = 300
17
18  #Spectra options
19  PS_type = 1
20  PS_version = 1
21
22  #GWs
23  GWprojectorType = 2
24  withGWs=true
25
26  #IC
27  kCutOff = 4
28  initial_amplitudes = 5.6964e18 0  # homogeneous amplitudes in GeV
29  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`

```
13  #Times
14  tOutputFreq = 1
15  tOutputInfreq = 1
16  tMax = 300
17
18  #Spectra options
19  PS_type = 1
20  PS_version = 1
21
22  #GWs
23  GWprojectorType = 2
24  withGWs=true
25
26  #IC
27  kCutOff = 4
28  initial_amplitudes = 5.6964e18 0  # homogeneous amplitudes in GeV
29  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	$\Omega_{\rm GW}$	Multiplicity
0.4	3.455976932183128e-29	13
0.8	1.13549488356703e-28	37
1.2	1.65395001818171 4e-28	57
⋮	⋮	⋮

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

We get two different output files

1. `spectra_gws.txt` → $\dfrac{1}{\tilde{\rho}_{\rm tot}}\dfrac{{\rm d}\tilde{\rho}_{\rm GW}}{{\rm d}\log k}$

κ	$\Omega_{\rm GW}$	Multiplicity
0.4	3.455976932183128e-29	13
0.8	1.13549488356703e-28	37
1.2	1.65395001818171 4e-28	57
⋮	⋮	⋮

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

We get two different output files

1. `spectra_gws.txt` → $\dfrac{1}{\tilde{\rho}_{\rm tot}}\dfrac{\mathrm{d}\tilde{\rho}_{\rm GW}}{\mathrm{d}\log k}$

κ	$\Omega_{\rm GW}$	Multiplicity
0.4	3.455976932183128e-29	13
0.8	1.13549488356703e-28	37
1.2	1.65395001818171 4e-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	1.2669487198562e-27	5.2978770016331 1e-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

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

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

src/include/CosmoInterface/evolvers/leapfrog.h

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

```
134 template<class Model>
135 void kickGWs(Model& model, T w) {
136     ForLoop(n, 0, Model::NGWs - 1,
137             (*model.piGWs)(n) += (w * model.dt) *
                ↪  GWsKernels::get(model,n);
138     );
139 }
```

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

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

src/include/CosmoInterface/evolvers/leapfrog.h

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

```
134 template<class Model>
135 void kickGWs(Model& model, T w) {
136     ForLoop(n, 0, Model::NGWs - 1,
137             (*model.piGWs)(n) += (w * model.dt) *
                ↪  GWsKernels::get(model,n);
138     );
139 }
```

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

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

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

src/include/CosmoInterface/evolvers/leapfrog.h

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

```
134 template<class Model>
135 void kickGWs(Model& model, T w) {
136     ForLoop(n, 0, Model::NGWs - 1,
137             (*model.piGWs)(n) += (w * model.dt) *
                ↪  GWsKernels::get(model,n);
138     );
139 }
```

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

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

```
201 template<class Model>
202 void driftGWs(Model& model) {
203
204     (*model.fldGWs) +=  pow(model.aSI, model.alpha - 3) * (model.dt *
        ↪  (*model.piGWs) );
205
206 }
```

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

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

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

```
30          template <class Model, int N>
31          static auto get(Model& model, Tag<N> n){
32
33              return (pow(model.aI, 1 + model.alpha) *
34                      LatLapl<Model::NDim>( (*model.fldGWs)(n))
35                       + pow(model.aI, 1 + model.alpha) * 2 *
                         ↪  pow<2>(model.fStar/Constants::MPl) *
                         ↪  (PITensor::totalTensor(model,n)));
36          }
```

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

GWs in CosmoLattice: $\mathcal{K}_{ij}[u_{ij}, \{\tilde{\phi}\}, a]$

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

```
30          template <class Model, int N>
31          static auto get(Model& model, Tag<N> n){
32
33              return (pow(model.aI, 1 + model.alpha) *
34                      LatLapl<Model::NDim>( (*model.fldGWs)(n))
35                        + pow(model.aI, 1 + model.alpha) * 2 *
                          ↪  pow<2>(model.fStar/Constants::MPl) *
                          ↪  (PITensor::totalTensor(model,n)));
36          }
```

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

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

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

```
30          template <class Model, int N>
31          static auto get(Model& model, Tag<N> n){
32
33              return (pow(model.aI, 1 + model.alpha) *
34                      LatLapl<Model::NDim>( (*model.fldGWs)(n))
35                        + pow(model.aI, 1 + model.alpha) * 2 *
                          ↪  pow<2>(model.fStar/Constants::MPl) *
                          ↪  (PITensor::totalTensor(model,n)));
36          }
```

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

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

src/include/CosmoInterface/definitions/PITensor.h

```
39      template<class Model>
40      static inline auto totalTensor(Model& model, Tag<0>)
41      {
42          return scalarSinglet(model, 1_c, 1_c) + complexScalar(model, 1_c, 1_c);
43      }
44      template<class Model>
45      static inline auto totalTensor(Model& model, Tag<1>)
46      {
47          return scalarSinglet(model, 1_c, 2_c) + complexScalar(model, 1_c, 2_c);;
48      }
```

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

src/include/CosmoInterface/definitions/PITensor.h

```
39      template<class Model>
40      static inline auto totalTensor(Model& model, Tag<0>)
41      {
42          return scalarSinglet(model, 1_c, 1_c) + complexScalar(model, 1_c, 1_c);
43      }
44      template<class Model>
45      static inline auto totalTensor(Model& model, Tag<1>)
46      {
47          return scalarSinglet(model, 1_c, 2_c) + complexScalar(model, 1_c, 2_c);;
48      }
```

```
80      template<class Model, int I, int J>
81      static inline auto scalarSinglet(Model& model, Tag<I> a, Tag<J> b)
82      {
83          return Total(i, 0, Model::Ns - 1, forwDiff(model.fldS(i),a) *
        ↪  forwDiff(model.fldS(i),b));
84      }
```

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

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

src/include/CosmoInterface/definitions/GWsProjector.h

```
49  template<class Model, class Looper, int I, int J, typename T = double>
50  inline auto Pr(Model& model, Looper& it, Tag<I> i, Tag<J> j) {
51
52      auto pVec = it.getVec();
53
54      size_t N = GetNGrid::get(model.getOneField());
55
56      auto klattice = MakeVector(l, 1, Model::NDim,
57          mType == 1 ? std::sin(2 * Constants::pi<T> * pVec[l-1] / N) :
58          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])) :
59          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.));
60
61      T klatticeSquare = Total(k, 1, Model::NDim,
        ↪  abs((klattice(k))*conj(klattice(k))));
62
63      if(i == j) return (std::complex<T>)(1) - conj(klattice(i)) * (klattice(j)) /
        ↪  klatticeSquare;
64      return - conj(klattice(i)) * (klattice(j)) / klatticeSquare;
```

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

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

src/include/CosmoInterface/definitions/GWsProjector.h

```
75  template<class Model, class Looper>
76  auto projectedGW_complex(Model& model, Looper& it) {
77
78      auto P11 = Pr(model, it, 1_c, 1_c);
79      auto P12 = Pr(model, it, 1_c, 2_c);
80      auto P13 = Pr(model, it, 1_c, 3_c);
81      auto P21 = conj(P12);
82      auto P22 = Pr(model, it, 2_c, 2_c);
83      auto P23 = Pr(model, it, 2_c, 3_c);
84      auto P31 = conj(P13);
85      auto P32 = conj(P23);
86      auto P33 = Pr(model, it, 3_c, 3_c);
87
88      auto u11 = GetValue::get(model.pi_GWtensor(1_c,1_c).inFourierSpace(), it());
89      auto u12 = GetValue::get(model.pi_GWtensor(1_c,2_c).inFourierSpace(), it());
90      auto u13 = GetValue::get(model.pi_GWtensor(1_c,3_c).inFourierSpace(), it());
91      auto u21 = u12;
92      auto u22 = GetValue::get(model.pi_GWtensor(2_c,2_c).inFourierSpace(), it());
93      auto u23 = GetValue::get(model.pi_GWtensor(2_c,3_c).inFourierSpace(), it());
94      auto u31 = u13;
95      auto u32 = u23;
96      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

```
75  template<class Model, class Looper>
76  auto projectedGW_complex(Model& model, Looper& it) {
77
78      auto P11 = Pr(model, it, 1_c, 1_c);
79      auto P12 = Pr(model, it, 1_c, 2_c);
80      auto P13 = Pr(model, it, 1_c, 3_c);
81      auto P21 = conj(P12);
82      auto P22 = Pr(model, it, 2_c, 2_c);
83      auto P23 = Pr(model, it, 2_c, 3_c);
84      auto P31 = conj(P13);
85      auto P32 = conj(P23);
86      auto P33 = Pr(model, it, 3_c, 3_c);
87
88      auto u11 = GetValue::get(model.pi_GWtensor(1_c,1_c).inFourierSpace(), it());
89      auto u12 = GetValue::get(model.pi_GWtensor(1_c,2_c).inFourierSpace(), it());
90      auto u13 = GetValue::get(model.pi_GWtensor(1_c,3_c).inFourierSpace(), it());
91      auto u21 = u12;
92      auto u22 = GetValue::get(model.pi_GWtensor(2_c,2_c).inFourierSpace(), it());
93      auto u23 = GetValue::get(model.pi_GWtensor(2_c,3_c).inFourierSpace(), it());
94      auto u31 = u13;
95      auto u32 = u23;
96      auto u33 = GetValue::get(model.pi_GWtensor(3_c,3_c).inFourierSpace(), it());
```

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

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

src/include/CosmoInterface/definitions/GWsProjector.h

```
75  template<class Model, class Looper>
76  auto projectedGW_complex(Model& model, Looper& it) {
77
78      auto P11 = Pr(model, it, 1_c, 1_c);
79      auto P12 = Pr(model, it, 1_c, 2_c);
80      auto P13 = Pr(model, it, 1_c, 3_c);
81      auto P21 = conj(P12);
82      auto P22 = Pr(model, it, 2_c, 2_c);
83      auto P23 = Pr(model, it, 2_c, 3_c);
84      auto P31 = conj(P13);
85      auto P32 = conj(P23);
86      auto P33 = Pr(model, it, 3_c, 3_c);
87
88      auto u11 = GetValue::get(model.pi_GWtensor(1_c,1_c).inFourierSpace(), it());
89      auto u12 = GetValue::get(model.pi_GWtensor(1_c,2_c).inFourierSpace(), it());
90      auto u13 = GetValue::get(model.pi_GWtensor(1_c,3_c).inFourierSpace(), it());
91      auto u21 = u12;
92      auto u22 = GetValue::get(model.pi_GWtensor(2_c,2_c).inFourierSpace(), it());
93      auto u23 = GetValue::get(model.pi_GWtensor(2_c,3_c).inFourierSpace(), it());
94      auto u31 = u13;
95      auto u32 = u23;
96      auto u33 = GetValue::get(model.pi_GWtensor(3_c,3_c).inFourierSpace(), it());
```

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

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

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

src/include/CosmoInterface/definitions/GWsProjector.h

```
98      auto Pu11 = P11 * u11 + P12 * u21 + P13 * u31;
99      auto Pu12 = P11 * u12 + P12 * u22 + P13 * u32;
100     auto Pu13 = P11 * u13 + P12 * u23 + P13 * u33;
101     auto Pu21 = P21 * u11 + P22 * u12 + P23 * u13;
102     auto Pu22 = P21 * u12 + P22 * u22 + P23 * u23;
103     auto Pu23 = P21 * u13 + P22 * u23 + P23 * u33;
104     auto Pu31 = P31 * u11 + P32 * u21 + P33 * u13;
105     auto Pu32 = P31 * u12 + P32 * u22 + P33 * u23;
106     auto Pu33 = P31 * u13 + P32 * u23 + P33 * u33;
107
108     auto Pu_s11 = P11 * conj(u11) + P12 * conj(u21) + P13 * conj(u31);
109     auto Pu_s12 = P11 * conj(u12) + P12 * conj(u22) + P13 * conj(u32);
110     auto Pu_s13 = P11 * conj(u13) + P12 * conj(u23) + P13 * conj(u33);
111     auto Pu_s21 = P21 * conj(u11) + P22 * conj(u12) + P23 * conj(u13);
112     auto Pu_s22 = P21 * conj(u12) + P22 * conj(u22) + P23 * conj(u23);
113     auto Pu_s23 = P21 * conj(u13) + P22 * conj(u23) + P23 * conj(u33);
114     auto Pu_s31 = P31 * conj(u11) + P32 * conj(u21) + P33 * conj(u13);
115     auto Pu_s32 = P31 * conj(u12) + P32 * conj(u22) + P33 * conj(u23);
116     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

```
98      auto Pu11 = P11 * u11 + P12 * u21 + P13 * u31;
99      auto Pu12 = P11 * u12 + P12 * u22 + P13 * u32;
100     auto Pu13 = P11 * u13 + P12 * u23 + P13 * u33;
101     auto Pu21 = P21 * u11 + P22 * u12 + P23 * u13;
102     auto Pu22 = P21 * u12 + P22 * u22 + P23 * u23;
103     auto Pu23 = P21 * u13 + P22 * u23 + P23 * u33;
104     auto Pu31 = P31 * u11 + P32 * u21 + P33 * u13;
105     auto Pu32 = P31 * u12 + P32 * u22 + P33 * u23;
106     auto Pu33 = P31 * u13 + P32 * u23 + P33 * u33;
107
108     auto Pu_s11 = P11 * conj(u11) + P12 * conj(u21) + P13 * conj(u31);
109     auto Pu_s12 = P11 * conj(u12) + P12 * conj(u22) + P13 * conj(u32);
110     auto Pu_s13 = P11 * conj(u13) + P12 * conj(u23) + P13 * conj(u33);
111     auto Pu_s21 = P21 * conj(u11) + P22 * conj(u12) + P23 * conj(u13);
112     auto Pu_s22 = P21 * conj(u12) + P22 * conj(u22) + P23 * conj(u23);
113     auto Pu_s23 = P21 * conj(u13) + P22 * conj(u23) + P23 * conj(u33);
114     auto Pu_s31 = P31 * conj(u11) + P32 * conj(u21) + P33 * conj(u13);
115     auto Pu_s32 = P31 * conj(u12) + P32 * conj(u22) + P33 * conj(u23);
116     auto Pu_s33 = P31 * conj(u13) + P32 * conj(u23) + P33 * conj(u33);
```

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

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

src/include/CosmoInterface/definitions/GWsProjector.h

```
98      auto Pu11 = P11 * u11 + P12 * u21 + P13 * u31;
99      auto Pu12 = P11 * u12 + P12 * u22 + P13 * u32;
100     auto Pu13 = P11 * u13 + P12 * u23 + P13 * u33;
101     auto Pu21 = P21 * u11 + P22 * u12 + P23 * u13;
102     auto Pu22 = P21 * u12 + P22 * u22 + P23 * u23;
103     auto Pu23 = P21 * u13 + P22 * u23 + P23 * u33;
104     auto Pu31 = P31 * u11 + P32 * u21 + P33 * u13;
105     auto Pu32 = P31 * u12 + P32 * u22 + P33 * u23;
106     auto Pu33 = P31 * u13 + P32 * u23 + P33 * u33;
107
108     auto Pu_s11 = P11 * conj(u11) + P12 * conj(u21) + P13 * conj(u31);
109     auto Pu_s12 = P11 * conj(u12) + P12 * conj(u22) + P13 * conj(u32);
110     auto Pu_s13 = P11 * conj(u13) + P12 * conj(u23) + P13 * conj(u33);
111     auto Pu_s21 = P21 * conj(u11) + P22 * conj(u12) + P23 * conj(u13);
112     auto Pu_s22 = P21 * conj(u12) + P22 * conj(u22) + P23 * conj(u23);
113     auto Pu_s23 = P21 * conj(u13) + P22 * conj(u23) + P23 * conj(u33);
114     auto Pu_s31 = P31 * conj(u11) + P32 * conj(u21) + P33 * conj(u13);
115     auto Pu_s32 = P31 * conj(u12) + P32 * conj(u22) + P33 * conj(u23);
116     auto Pu_s33 = P31 * conj(u13) + P32 * conj(u23) + P33 * conj(u33);
```

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

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

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

src/include/CosmoInterface/definitions/GWsProjector.h

```
118     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);
119     auto Tr2a = (Pu11 + Pu22 + Pu33);
120     auto Tr2b = (Pu_s11 + Pu_s22 + Pu_s33);
121
122     return pow(model.aI,2*(model.alpha-3)) * abs(Tr1 - .5 * Tr2a *
        ↪  Tr2b);
123 }
```

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

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

src/include/CosmoInterface/definitions/GWsProjector.h

```
118     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);
119     auto Tr2a = (Pu11 + Pu22 + Pu33);
120     auto Tr2b = (Pu_s11 + Pu_s22 + Pu_s33);
121
122     return pow(model.aI,2*(model.alpha-3)) * abs(Tr1 - .5 * Tr2a *
        ↪  Tr2b);
123 }
```

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

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

src/include/CosmoInterface/definitions/GWsProjector.h

```
118     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);
119     auto Tr2a = (Pu11 + Pu22 + Pu33);
120     auto Tr2b = (Pu_s11 + Pu_s22 + Pu_s33);
121
122     return pow(model.aI,2*(model.alpha-3)) * abs(Tr1 - .5 * Tr2a *
        ↪  Tr2b);
123 }
```

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

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

src/include/CosmoInterface/definitions/GWsProjector.h

```
118     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);
119     auto Tr2a = (Pu11 + Pu22 + Pu33);
120     auto Tr2b = (Pu_s11 + Pu_s22 + Pu_s33);
121
122     return pow(model.aI,2*(model.alpha-3)) * abs(Tr1 - .5 * Tr2a *
        ↪  Tr2b);
123 }
```

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

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

src/include/CosmoInterface/definitions/GWsProjector.h

```
118     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);
119     auto Tr2a = (Pu11 + Pu22 + Pu33);
120     auto Tr2b = (Pu_s11 + Pu_s22 + Pu_s33);
121
122     return pow(model.aI,2*(model.alpha-3)) * abs(Tr1 - .5 * Tr2a *
        ↪  Tr2b);
123 }
```

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

Similar for the real projector

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

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

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

see Technical note I: Power Spectrum

```
54      if (PSVersion != 3){
55          auto fk2 = projectRadiallyGW(model, PSVersion == 1, PRJType, PSType,
            ↪  PSVersion).measure(model, nbins, kMaxBins);
56          if (PSType == 2){
57              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;
58          }
```

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

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

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

see Technical note I: Power Spectrum

```
54      if (PSVersion != 3){
55          auto fk2 = projectRadiallyGW(model, PSVersion == 1, PRJType, PSType,
            ↪  PSVersion).measure(model, nbins, kMaxBins);
56          if (PSType == 2){
57              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;
58          }
```

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

```
59          else if (PSType == 1){
60              fk2.sumInsteadOfAverage();
61              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;
62          }
```

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

Working Examples

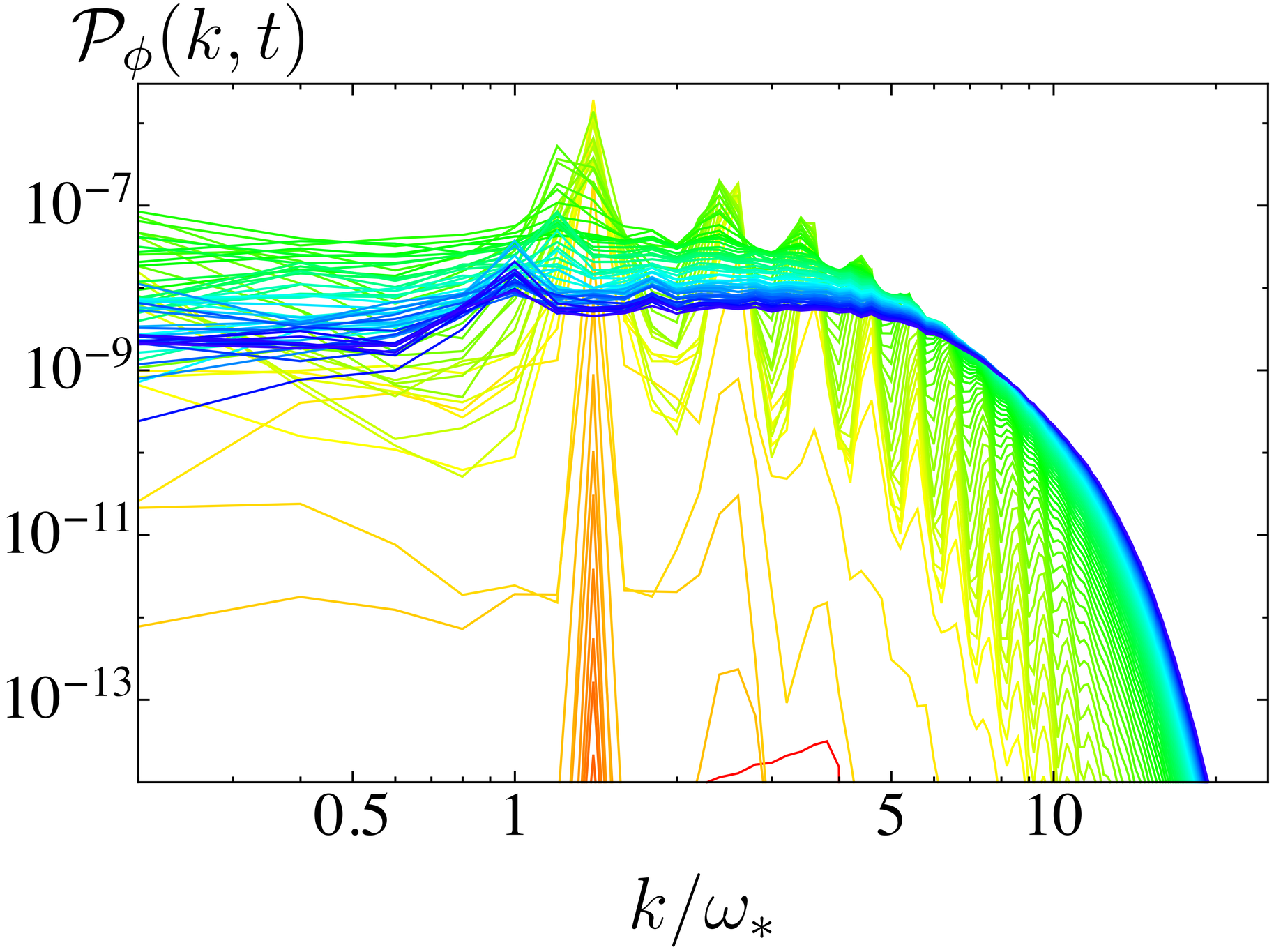
$\lambda\phi^4$ Self resonance

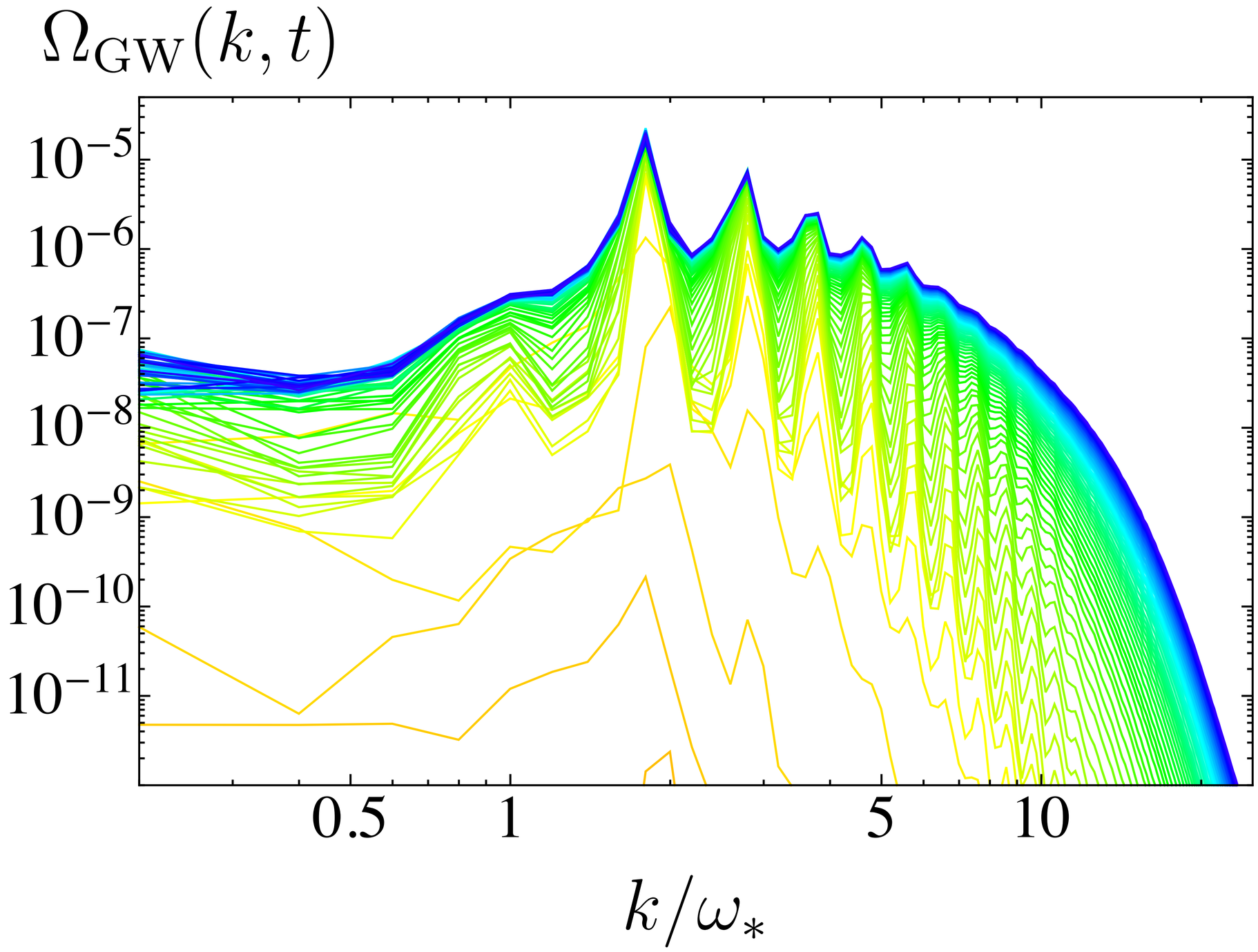
```
#Evolution
expansion = true
evolver = LF

#Lattice
N = 64
dt = 0.05
kIR = 0.5

#Times
tOutputFreq = 5
tOutputInfreq = 5
tMax = 1500

#Power spectrum options
PS_type = 1
PS_version = 1

#GWs
GWprojectorType = 1
withGWs=true

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

#Model Parameters
lambda = 9e-14
```

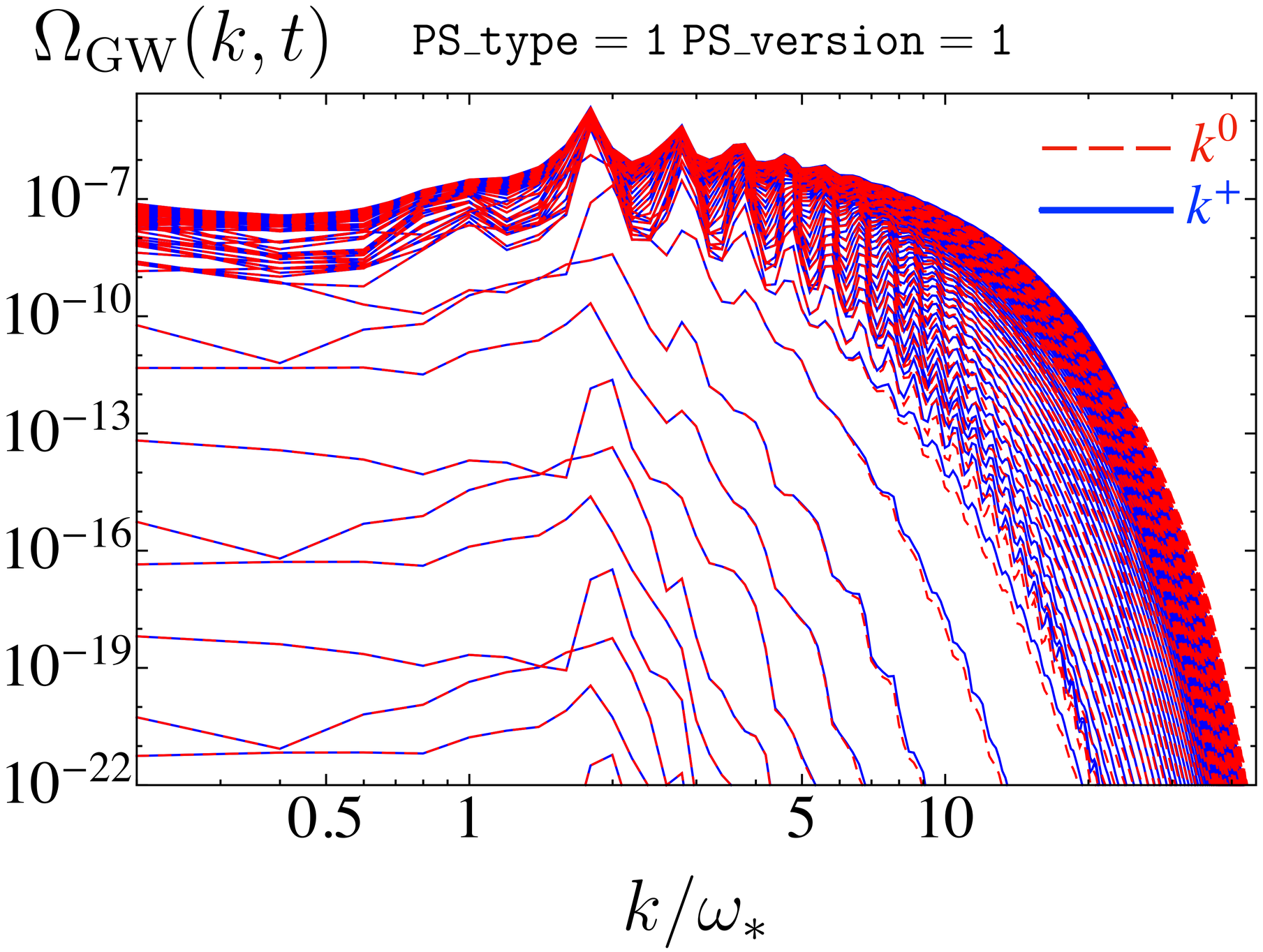
Write the model

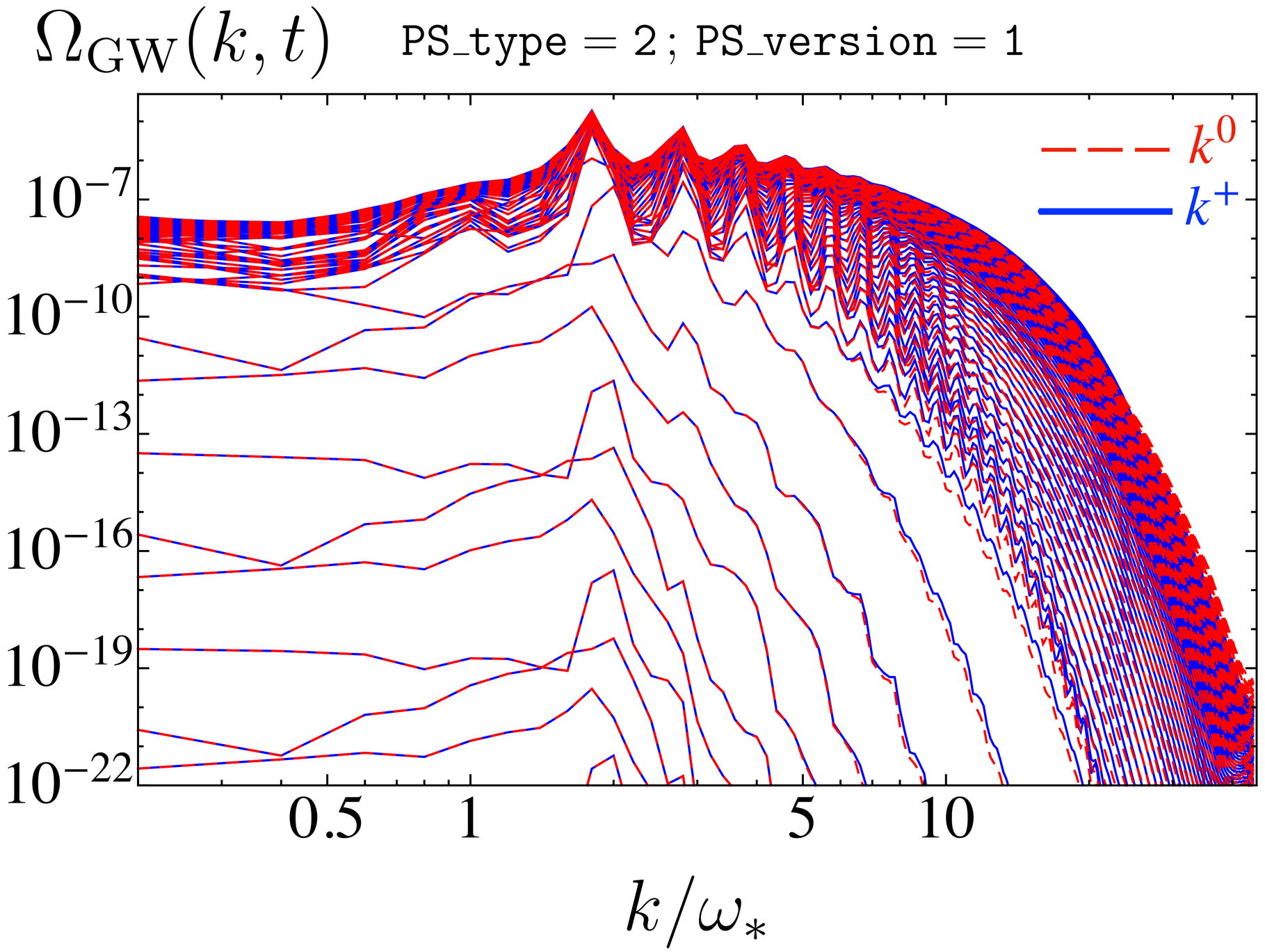
$$V(\phi) = \frac{1}{4}\lambda\phi^4$$

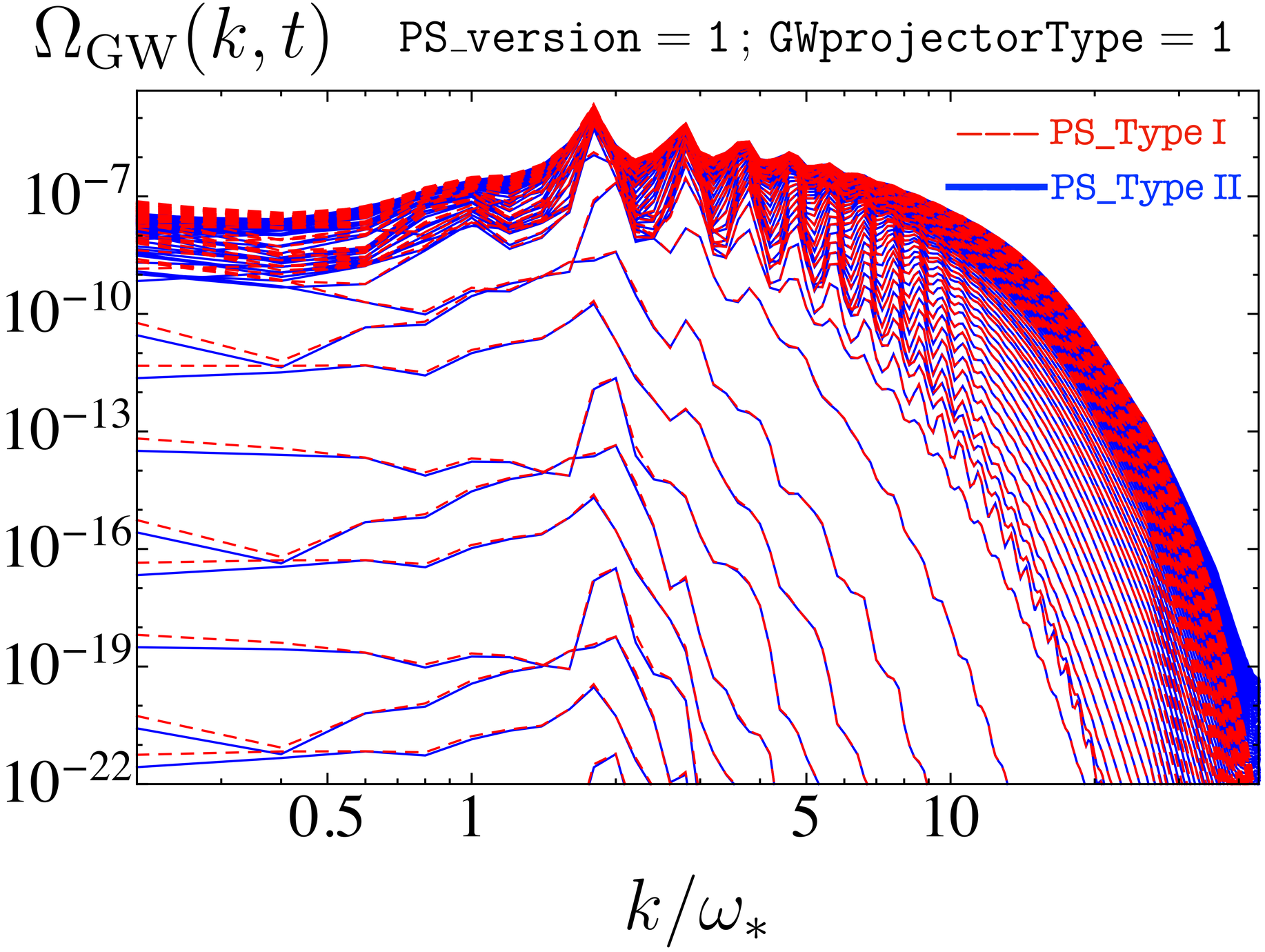
$$\ddot{\phi} + 3H\dot{\phi} + \lambda\phi^3 = 0$$

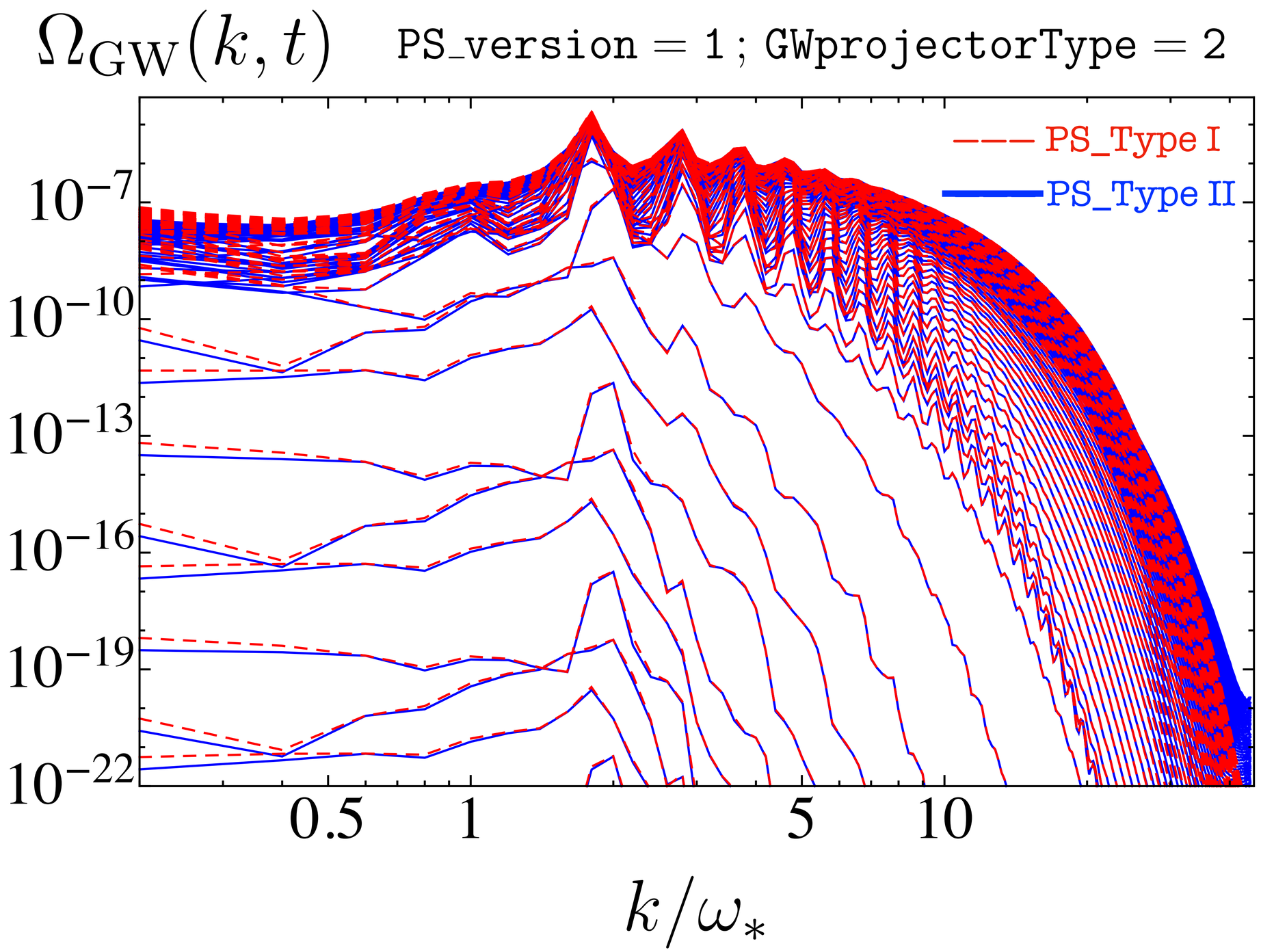

$\mathcal{P}_\phi(k,t)$
10^{-7}
10^{-9}
10^{-11}
10^{-13}
0.5
1
5
10
k/ω_*


$\Omega_{\rm GW}(k,t)$
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{-9}
10^{-10}
10^{-11}
0.5
1
5
10
k/ω_*


$\Omega_{\rm GW}(k,t)$
PS_type = 1 PS_version = 1
k^0
k^+
10^{-7}
10^{-10}
10^{-13}
10^{-16}
10^{-19}
10^{-22}
0.5
1
5
10
k/ω_*


$\Omega_{\rm GW}(k,t)$
PS_type = 2; PS_version = 1
k^0
k^+
10^{-7}
10^{-10}
10^{-13}
10^{-16}
10^{-19}
10^{-22}
0.5
1
5
10
k/ω_*


$\Omega_{\rm GW}(k,t)$
PS_version = 1 ; GWprojectorType = 1
PS_Type I
PS_Type II
10^{-7}
10^{-10}
10^{-13}
10^{-16}
10^{-19}
10^{-22}
0.5
1
5
10
k/ω_*


$\Omega_{\rm GW}(k,t)$
PS_version = 1 ; GWprojectorType = 2
PS_Type I
PS_Type II
10^{-7}
10^{-10}
10^{-13}
10^{-16}
10^{-19}
10^{-22}
0.5
1
5
10
k/ω_*