Contents

Part I

Latest

1 Introduction

The simulation documented here is based on 3D density fluctuations in a homogeneous medium. Especially plane wave-packets are implemented as a basic event-type. A large amount of wave events can be generated by `CreateDataSet2.py`. It will calculate a time series of the 1D-Newtonian noise force caused by such wave events at an arbitrary number of points with adaptable, but fixed positions and projections. The wave events and their forces will be saved. They can be read again with `LoadNOptimize2.py` or `LoadNVisualize2.py`, where windows can be created. A window can contain multiple wave events and additional noise. The displacement field at arbitrary positions for an arbitrary mix of P- and S-waves can be calculated. `LoadNVisualize2.py` is supposed to produce descriptive plots that illustrate the wave events. `LoadNOptimize2.py` contains an implementation of the Wiener filter to evaluate the prediction power of given seismometer positions for some frequency in the form of the Newtonian noise residual. It can be used to iteratively optimize these positions.

Here is a very shortened overview of the code's workings:

`CreateDataSet2.py`:

for number of wave events **do**:
    draw wave event parameters
    for number of time steps **do**:
        calculate density fluctuations
        for number of mirrors **do**:
            calculate force
save parameters and force

`LoadNOptimize2.py` - Residual:

load parameters and force
for number of windows **do**:
    for number of wave events per window **do**
        calculate displacement field at given state
        add wave event to window
    if training phase **then**

calculate Fourier transform and CPSDs
calculate Wiener filter
else
calculate residuals
return mean residual

2 Dataset Creation

This describes the workings of `CreateDataSet.py`.

2.1 Dataset Structure

2.1.1 Settings

The dataset settings are summarized in `settingFileX.txt` where `X` is the name of the dataset. It consists of pairs written like 'variable = value' and, at the very end, a time stamp that documents the run-time of the dataset creation process.

2.1.2 Wave events

The parameters describing a wave event are saved in files named `wave_event_data_X.npy`. They contain `NoR` floats each. One for each event.
Saved parameters are the wave direction decoded by two angles `polar_angles` (φ) and `azimuthal_angles` (ϑ) given in rad, the wave amplitude `A` (A) in units of TODO, the reference time and position `t0` (t_0) in seconds and `x0` (x_0) in m (which are usually not modified), the frequency and frequency width of the wave packet `f0` (f_0) and `sigmaf` (σ_f) in Hz and potentially a polarization angle `s_polarization` (α_S) in rad that determines the direction of polarization if it is classified as an S-wave in the evaluation.

2.1.3 Forces

The calculated force is saved in i files named `wave_event_result_force_i_X.npy`. They contain N_t floats: the force in Newton for each time step.

2.2 Settings

Basically every quantity is given in SI-units.

2.2.1 Save Management

- `ID`
  An identifier if multiple datasets with the same name are produced, e.g. by HTCondor.
- `tag`
  A name for the dataset that should be unique
- `folder`
  The path where to save the dataset starting from the directory of `CreateDataSet2.py`. Omit the last '/'. Set an empty string if you want to save the files in the same folder as `CreateDataSet2.py`.

2.2.2 General

- `useGPU`
  Set to False on CPU, set to True if you want to calculate on GPU resources.
- `randomSeed`
  If 'None', the results are unreproducibly random. If you want reproducability, set to some number.
- `isMonochromatic`
  One can choose between generating monochromatic plane waves of a single frequency and Gaussian wave packets by setting this boolean
- `NoR`
  Number of runs/realizations/wave events. This determines how many events are contained in the dataset.

2.2.3 Constants

- `G`
  The gravitational constant $G = 6.6743 \cdot 10^{-11}\,\mathrm{m}^3/\mathrm{s}^2/\mathrm{kg}$.

- `M`
  This is the mass M of the mirrors set to 211 kg.

- `rho`
  The density ρ of the homogeneous rock in kg/m^3. Default is 3000 kg/m^3.

- `c_p`
  The sound speed c_P of rock assumed to be constant. Set to 6000 m/s.

2.2.4 Domain

- `L`
  The radius L of the spherical integration boundary to calculate the force. Units are meter.

- `Nx`
  The number of spatial steps/gridpoints N_x in one dimension of the cube surrounding the integration domain (the density fluctuations are defined in a 3D-array which is cubic by nature). This should be an even number.

- `dx`
  The distance Δx between two grid points in the integration domain.

- `tmax/xmax`
  The length of the simulation $t_{\max}$ in seconds is currently derived from $2x_{\max}/c_P$, where $x_{\max} = 2L$ is the distance from 0 in meters at which the center of the wave is at time $t = 0$. All x_0 of the wave events are set to $-x_{\max}$ and all t_0 are set to 0. Changes in these parameters will be artificially introduced by adding them to windows during evaluation.

- `Nt`
  The number of time steps N_t between 0 and $t_{\max}$.

- `dt`
  The distance between two time steps Δt given in seconds.

2.2.5 Experiment Configuration

- `cavity_r`
  The radius r_{c} of the spherical cavities assumed around every mirror.

They will be cut out from the integration domain.

- `mirror_positions`
  A list of 3D vectors $\vec{x}_M$ giving the positions of mirrors in meter. The force at all of these positions will be evaluated.

- `mirror_directions`
  A list of 3D unit vectors $\vec{e}_M$ that should match the mirror positions. They describe the axis along which the mirror can move freely. The force at each mirror position will be projected onto this direction.

- `mirror_count`
  The number of mirrors considered. It is also the number of forces calculated and is derived directly from the number of entries of `mirror_positions`.

2.2.6 Event Parameters

- `Awavemin/Awavemax`
  The range of amplitudes A the wave events can have. They are drawn from a equal distribution between `Awavemin` and `Awavemax`

- `fmin/fmax`
  The range of frequencies f_0 the wave events are drawn from, if they are not monochromatic. Again, drawn from an equal distribution.

- `fmono`
  Is used for f_0 if `isMonochromatic`. All wave events then carry this frequency.

- `sigmafmin/sigmafmax`
  The range of frequency widths σ_f the wave packets can have if not `isMonochromatic`. Equally distributed.

The default for the generation of wave events is a uniform distribution, but this can be changed.

2.3 Calculations

The parameters for `NoR` wave events are drawn, typically from uniform distributions between a maximum and a minimum value (see Table 1). It is

incident polar angle	φ	$[0, 2\pi)$
incident azimuthal angle	$\cos(\vartheta)$	$[0, 1)$
amplitude	A	$[A_{\min}, A_{\max})$
phase	ϕ	0
frequency	f_0	$[f_{\min}, f_{\max})$ or f_{mono}
frequency width	σ_f	$[\sigma_{f,\min}, \sigma_{f,\max})$ or 10^{-10}
polarization angle	α_S	$[0, 2\pi)$

Table 1: Default parameter intervals

also possible to create datasets with different distributions of wave event parameters, e.g. anisotropic.

The time dimension t is given by a 1D-array with N_t entries from 0 to $t_{\max}$. The integration domain $\vec{x}$ is defined as a 3D-grid with spacing Δx and N_x gridpoints per dimension x, y and z. It stretches from $-L+\Delta x/2$ to $L-\Delta x/2$. A mask $K(\vec{x})$ will conceal all contributions, where $r = \sqrt{x^2+y^2+z^2} > L$, so that a spherical boundary is applied. This is important because for monochromatic plain waves, the contribution of the outer volume will not converge to 0 for large integration boundaries. (Technically, an integration over an infinite rock volume is ill-defined and the contribution of an outer surface must always be considered. But in reality, the contribution of the far rock will most likely average to 0, which spherical boundaries also satisfy.) Furthermore, the mask cuts out a small radius of r_{c} around each of the mirror positions $\vec{x}_M$. The geometric factors on the domain for the projection of each mirror are precalculated.

The plane wave-packet density fluctuations defined on the integration domain are given by

$$\delta\rho(\vec{x}, t)/\rho = A e^{2\pi^2\sigma_f^2 t'^2} \cos\big(2\pi f_0 t' + \phi\big) \tag{1}$$

where

$$t' = (\vec{x} \cdot \vec{e}_{\text{wave}} - x_0)/c_P - (t - t_0) \tag{2}$$

where $\vec{e}_{\text{wave}}$ is the direction, the wave is propagating along, defined by φ and ϑ. This formula may also be used for plane waves, but here f_0 will be fixed and σ_f will be set to a very small value (See Table 1).

The Newtonian noise force of a test mass M can be found by integration and projection onto an axis:

$$F_M(t) = GM \int \mathrm{d}\vec{x}\, \frac{(\vec{x} - \vec{x}_M) \cdot \vec{e}_M}{|\vec{x} - \vec{x}_M|^3} \delta\rho(\vec{x}, t) K(\vec{x}) \tag{3}$$

This integration will be performed for each time step for all mirrors.

In the last step, all the produced data is saved according to the file structure in Section 2.1.

Be careful to check, whether the volume of integration is large enough. For monochromatic wavelengths, it should have a size of several wavelengths, so that the outer volume does not yield a significant contribution to the force.

3 Dataset Evaluation

The code of `LoadNOptimize.py` and `LoadNVisualize.py` is the same in large amounts. Therefore in summary, they will be called `LoadNX.py`

3.1 Settings

3.1.1 Load and Save Management

- `ID`
  An identifier if multiple datasets with the same name are produced, e.g. by HTCondor.

- `tag`
  The name of some existing dataset.

- `folder`
  The path where the dataset is starting from the directory of `LoadNX.py`. Omit the last '/'. Set an empty string if you want to save the files in the same folder as `LoadNX.py`.

- `saveas`
  Where to save the results of the evaluation and optimization. Can be used to adapt the name of the save files.

3.1.2 General

- `useGPU`
  Set to False on CPU, set to True if you want to calculate on GPU resources.

- `NoR`
  Number of runs/realizations/wave events. With this, you can read a part of the dataset only. If it is bigger than the size of the dataset, the complete dataset will be loaded.

3.1.3 Window Options

- `useGPU`
  Set to False on CPU, set to True if you want to calculate on GPU resources.

- `NoR`
  Number of runs/realizations/wave events. This determines how many events are contained in the dataset.

3.1.4 Parameters

- `state`
  To specify a list of seismometer positions $\vec{x}_S$ in 3D.

- `NoS`
  Number of seismometers N.

- `freq`
  Frequency f the Wiener filter will be optimized on.

- `SNR`
  The level of noise SNR in frequency space as defined by Francesca.

- `p`
  The ratio between P- and S-waves p. $p = 1$ means P-waves only.

- `c_ratio`
  Ratio c_s/c_p.

3.1.5 Window parameters

- `NoE`
  The number of wave events to be placed in each time window. They will be added according to the principle of superposition.

- `NoW`
  The total number of time windows to be created. As long as `NoW`$\leqslant$`NoR`/`NoE`, the wave events will be used one by one. If the number is larger, the remaining time windows will use random wave events collected from the whole loaded dataset.

- `NoT`
  The number of time windows used in the test set and not for training of the Wiener filter.

- `time_window_multiplier/twindow`
  If `twindow=None`, the length of a window t_W is given by `time_window_multiplier`$\cdot t_{\max}$, where $t_{\max}$ is the time length of a wave event in the dataset. Otherwise, `twindow` should be a time t_W in seconds.

- `randomlyPlaced`
  Boolean that determines if wave events are shifted randomly in the window before they are added.

3.2 Calculations

The dataset is loaded. Parameters are imported from the setting file and some preparations are made.

3.2.1 Windows

A window is an object that reorganizes the single wave events from the dataset by allowing arbitrary overlap and lengths of the time window, possibly also a single large data stream to be evaluated. They contain a time series of the force at all mirrors in the experimental setup of the dataset, as well as the according displacement time series for a given states of seismometer positions.

Force and displacement given a wave event and seismometer positions will be calculated by a dedicated function called `getForceAndDisplacement`.

The displacement $\vec{\xi}$ is calculated by the following analytical expression gained from the integration of the density fluctuations following $\delta\rho/\rho = -\vec{\nabla}\cdot\vec{\xi}$ along the direction of propagation:

$$\xi = A'/2\left(\mathrm{erf}\left(\frac{2\pi\sigma_f^2 t' + \mathrm{i}f_0}{\sqrt{2}\sigma_f}\right) + e^{2i\phi}\mathrm{erf}\left(\frac{2\pi\sigma_f^2 t' - \mathrm{i}f_0}{\sqrt{2}\sigma_f}\right)\right) \tag{4}$$

or equivalently if the phase $\phi = 0$

$$\xi = A'\mathrm{Re}\left[\mathrm{erf}\left(\frac{2\pi\sigma_f^2 t' + \mathrm{i}f_0}{\sqrt{2}\sigma_f}\right)\right] \tag{5}$$

with

$$A' = -A\frac{c_P}{2\sqrt{2\pi}\sigma_f}e^{-\frac{f_0^2}{2\sigma_f^2} - \mathrm{i}\phi} \tag{6}$$

for Gaussian wave packets and, respectively,

$$\xi = A\frac{c_P}{2\pi f_0}\sin\big(2\pi f_0 t' + \phi\big) \tag{7}$$

for monochromatic waves. The value will be projected onto the x-, y- and z-axis to simulate broadband seismometers with 3 channels. For S-waves, it will be perpendicular to the direction of movement according to the polarization α_S.

S-waves will be treated differently from P-waves. While for P-waves, the saved force from the dataset will be the base for the Newtonian noise, for the randomly selected S-waves (according to p) this force will be 0. For both, an additional cavern term will be introduced as

$$\frac{4\pi}{3}GM\rho\vec{\xi}(\vec{x}_M, t) \tag{8}$$

which accounts for the Newtonian noise from the displacement of the spherical caverns around the mirrors (the displacements of the other caverns are ignored).

After the time window is finalized, noise can be added to the window's displacement series.

In `LoadNVisualize.py`, there are additional visualization functions to make windows more accessible.

3.2.2 Wiener Filter Residual

The function `frequencyWienerFilter` calculates the Wiener filter in frequency spaces that optimizes the prediction of Newtonian noise from given seismometer channels. It depends on the number and positions of the seismometers, the frequency, the SNR, the P-wave fraction and the specific mirror of the setup. It is trained on a subset of windows given by `NoT`. Windows are checked on their validity (NaN) and filtered.

The Wiener filter will be calculated from the training set samples of the fast Fourier transform of the force, which I call the signal s and the displacement data $\vec{d}$. s is a single value at the specified frequency f in Fourier space. $\vec{d}$ is a vector consisting of a single value for each of the $3N$ data channels recording the displacement.

The Wiener filter is calculated as

$$\vec{W}_F = \langle \vec{d}^* \vec{d} \rangle^{-1} \cdot \langle \vec{d}^* s \rangle \tag{9}$$

Additionally, the function returns the signal's power spectral density $\langle s * s \rangle$, the cross power spectral density of data and signal $\langle \vec{d}^* s \rangle$, the data cross power spectral density $\langle \vec{d}^* \vec{d} \rangle$ and its inverse in a dictionary.

The function `evaluateFrequencyWienerFilter` applies the Wiener filter to the test set, defined by `NoT` and evaluates its prediction abilities. It calculates, what is called the square root of the Residual $\sqrt{R}$.

3.2.3 Optimization and Visualization

The functions written in the files `LoadNX.py` can be used for further calculations and adaptions to different problems.

To take a look at the datastreams involved, create a window of your choice and use the built-in function `window.vizualizeWindow()` to receive a summarized animation or picture of the force, the displacements, the experimental setup and the density fluctuations at a timestep.

To optimize the seismometer positions, minimize the function `residual()` for a single mirror or `combinedResidual()` for multiple mirrors and set `add_info=False`.

To evaluate a given position with detail, also use these functions, but use `add_info=True`.

Part II

Old

4 Introduction

The simulation I have written is calculating the force on a mirror in x-direction and the x-, y- and z-displacement of arbitrarily placed seismometers based on wave-like density fluctuations. With these two information (data and signal) from a large number of simulation runs, I can calculate the Wiener filter in time or frequency domain to predict the signal from the data and quantify the residual.

Here is a very shortened overview of the code's workings:

for number of runs **do**:
    draw wave events
    for number of time steps **do**:
        calculate force
        calculate seismometer displacements
    if training phase **then**
        calculate Fourier transform and CPSDs
        calculate Wiener filter
    else
        calculate residuals

5 Parameter Overview

There are a couple of parameters that can be set. Some are physics parameters, some others influence the accuracy and time consumption of the simulation. This chapter is supposed to summarize all tunable parameters of the code.

5.1 Simulation Parameters

- $t_{\max}$
  The time up to each single simulation run extends in units of seconds. Default is $t_{\max} = 1$.

- N_t
  The number of timesteps the calculation is performed for. This will be the size of the time dimension with a time spacing of $\mathrm{d}t = \frac{t_{\max}}{N_t}$ from $t = 0$ to $t = t_{\max} - dt$. In order to calculate the residual of a certain Wiener filter frequency, be aware of the Nyquist frequency. Default is $N_t = 100$.
  The default values of $t_{\max}$ and N_t are suitable to calculate integer valued frequencies in Hz up to 50 Hz.

- L
  The base size of the 3D-Box that will contain the density fluctuations. It is the base for the grid-spacing for both the force and the displacement calculation, but is only a limit for the force calculation. The simulation box will range from $-L$ to L in x, y and z-direction and is measured in meter. Default is $L = 1000$.

- N_x
  This is the number of grid points for the calculation of the mirror force per dimension for the distance between the mirror at 0 and L. To calculate the force, an integral over the density fluctuation field is performed that is given at the $(2N_x)^3$ discrete grid points. The grid spacing, thus, will be $\mathrm{d}x = L/N_x$. Default is $N_x = 25$.

- $N_{x,\mathrm{dis}}$
  This mainly sets the accuracy of integration for the 1D-displacement calculation. The stepsize of integration will be $\mathrm{d}x_{\mathrm{dis}} = L/N_{x,\mathrm{dis}}$. The default value is $N_{x,\mathrm{dis}} = 10000$.

- `useSecondDerivative`
  If `True`, then the numerical acceleration of seismometers will be calculated instead of their displacement. The default is `False`.

- `NoE`
  The number of events that are included in each run. 'event' is refering to a single Gaussion wave packets or plane wave depending on the settings. Default is `NoE` = 10.

- `NoR`
  The total number of runs of the simulation. This will determine the quality of the Wiener filter as the errors on the averages decrease with increasing statistics. In principle, this could be parallelized. The default is `NoR` = 5000.

- `NoT`
  The number of runs that are not used for training the Wiener filter but for testing its ability to predict the Newtonian noise. The default is `NoT` = 500.

5.2 Physics Parameters

- M
  This is the mass of the mirror set to 211 kg.

- c_p
  The sound speed of rock assumed to be constant at this point. Set to 6000 m/s.

- r_{cavity}
  The radius of the cavity surrounding the mirror in meter, which assumed to be spherical. This is basically just volume that is ignored for the purpose of calculating the gravitational force on the mirror from the density fluctuations. This both simulates the missing rock in the vicinity of the mirror and prevents the integral from diverging. Default is $r_{\text{cavity}} = 5$.

- `NoS`
  The number of seismometers included in the simulations. Their positions in meter can either be set manually or randomly drawn. This is called N in the analytic calculation.

- `ID`
  Currently this is used as the frequency of plane waves for the Wiener filter calculation similar to the analytic calculation.

- SNR
  This is supposed to simulate the SNR-parameter from the analytic calculation but currently does not work properly.

- `isPlaneWave`
  Determines if there are plane waves or wave packets in the calculation. There are some more parameters for wave packets but I am too lazy to write about them now.

Currently, S-waves are not properly implemented, so it is always $p = 1$.

6 The Simulation

The code operates on two distinct 3-dimensional grids and on consecutively calculated time steps to propagate waves and calculate the force and the 3D-seismometer displacement as a function of time. In every run, NoE wave events are rolled and

6.1 Preparation

After parameters have been chosen and seismometer positions are decided, the simulation begins. Each run starts with a preparation, where event properties are drawn for all `NoE` events.

- Polar angles φ are drawn from a uniform distribution between 0 and 2π.
- Azitmuthal angles are drown from $\operatorname{acos} x$ where x is uniformly distributed between -1 and 1.
- x_0 is drawn uniformly between $-2L$ and $-L$. For wave packets, this is to prevent them from suddenly appearing inside the ??? does this really happen though?
- t_0 and x_0 basically just a phase for plane waves
- For the time being, it is always $A = 1$, which means, the density fluctuations are normalized to some value.

Before I start propagating the waves, I pre-calculate the displacement of a seismometer along the direction of each wave. This is just a 1D-integral and can be rotated later and adapted to the position of the seismometers. The displacement $\vec{\xi}(\vec{x}, t)$ and density fluctuations $\delta\rho(\vec{x}, t)$ are connected by the continuity equation:

$$\delta\rho(\vec{x}, t) = -\vec{\nabla} \cdot \left(\rho(\vec{x}, t)\vec{\xi}(\vec{x}, t)\right) \tag{10}$$

For a constant background density $\rho(\vec{x}, t) = \rho_0$ and only one relevant coordinate, it is easy to calculate the displacement by integration.

$$\xi(x, t) = \int_a^x \frac{\delta\rho(x', t)}{\rho_0} \mathrm{d}x' \tag{11}$$

where the lower border is just the choice for the reference point for the displacement and is not too important. For wave-packages, this can be simply chosen to be far away from the wave packets ($a \to -\infty$) to ensure a displacement of 0 when there is no wave present, but for plane waves, this is just chosen to be $a = 0$ as the wave extend is infinity. This introduces an offset for the displacement. Later, the absolute displacement of each seismometer can be achieved by adjusting the x-coordinate accordingly to its position.

6.2 Wave propagation

For each time step, the relative density fluctuation field $\frac{\delta\rho}{\rho}(\vec{x}, t)$ is calculated on a 3D-grid from $-L$ to L with N_x gridpoints in each dimension. These fluctuations are the sum of all wave events at time t and are calculated from the analytic representations of Gaussion wave packets. In the case of plane waves, the frequency width Δf is set to 10^{-10} Hz:

$$\begin{aligned}\frac{\delta\rho}{\rho}(\vec{x}, t) = \sqrt{2\pi} \cdot \Delta f \cdot A \cdot \exp\Big(-2\pi^2 \cdot \Delta f^2 \cdot (|\vec{x} - \vec{x}_0|/c_P - (t - t_0))^2\Big) \\ \cdot \sin\Big(2\pi \cdot f \cdot (|\vec{x} - \vec{x}_0|/c_P - (t - t_0)) + \phi\Big)\end{aligned} \tag{12}$$

where the phase $\phi = 0$ and all other parameters as described in Section 5 or Section 6.1.

As a next step, I integrate over this density fluctuation field to get the Newtonian noise at the mirror, which is located at the origin, in x-direction:

$$F_x(t) = GM \int_{|\vec{x}|>r_{\text{cavity}}} \mathrm{d}\vec{x} \cos(\varphi(\vec{x})) \sin(\vartheta(\vec{x})) \frac{\delta\rho(\vec{x}, t)}{|\vec{x}|^2} \tag{13}$$

where G is the gravitational constant, M the mass of the mirror and $\varphi(\vec{x})$ and $\vartheta(x)$ the polar and azimuthal angle of spherical coordinates expressed by the cartesian coordinates.

Also, the current displacement of each seismometer is read from the array of pre-calculated displacements for each event, projected onto the seismometer axes and summed up.

In the end, we have the force on the mirror $F(t)$ as the one dimensional signal and the displacements $\xi_{s,i}(t)$ of `NoS` seismometers in 3 dimensions as multidimensional data ($1 \leqslant s \leqslant N =$ `NoS` and $i = x, y, z$).

If `useSecondDerivative=True`, then now, each measured displacement is derived twice numerically.

Finally, Gaussian white noise is added by adding a Gaussian to each time step with a mean of $\mu = 0$ and a standard deviation of

$$\sigma = 10\sqrt{3}\sqrt{\langle(\max \xi_{s,i} - \min \xi_{s,i})/2\rangle}/\mathrm{SNR} \tag{14}$$

This is analogous to the analytic calculation of Francesca where $\mathrm{SNR}^2 = \frac{\langle\tilde{\xi}^*\tilde{\xi}\rangle}{\langle\tilde{n}^*\tilde{n}\rangle}$. The $\tilde{\cdot}$ denotes the Fourier transform and $\langle\tilde{n}^*\tilde{n}\rangle = \sigma^2$ is the white Gaussian noise PSD. The factor of $10\sqrt{3}$ is found experimentally to account for the discrepancy between analytical results and my simulation and cannot be explained yet. It might arise from the conversion from Fourier space.

6.3 Wiener filter

Given the signal and the data, the Wiener filter can be calculated and used to predict the signal from the data.

Fistly, both the force and the displacement are converted into Fourier space via a FFT. To be comparable with Francesca, now only one frequency f is considered for the Wiener filter. So we have only one value $s = \tilde{F}(f)$ and a data vector of length $3 \cdot \texttt{NoS}$ which contains $\vec{d} = \tilde{\xi}_{s,i}(f)$.

Then as long as we are in the first $\texttt{NoR} - \texttt{NoT}$ runs, the CPSDs for data and signal ($c_{ss} = \langle s^* s\rangle$, $\vec{c}_{sd} = \langle\vec{d}^* s\rangle$ and $C_{dd} = \langle\vec{d}^*\vec{d}\rangle$) are calculated and afterwards combined to the Wiener filter $\vec{W}_F = C_{dd}^{-1} \cdot \vec{c}_{sd}$. For the last `NoT` runs, I no longer update the expectation values but instead evaluate the performance of the Wiener filter. I calculate what I call the theoretical residual

$$R_{\mathrm{theo}} = 1 - \frac{\vec{c}_{sd}^* \cdot C_{dd}^{-1} \cdot \vec{c}_{sd}}{c_{ss}} \tag{15}$$

and the experimental residual from the mean error of the `NoT` runs

$$R_{\mathrm{exp}} = \frac{\left[\left|s - \vec{W}_F \cdot \vec{d}\right|^2\right]}{c_{ss}} \tag{16}$$

where $\langle\cdot\rangle$ denotes the mean over the training set and $[\cdot]$ denotes the mean over the test dataset.

If the Wiener filter has seen enough events during training, they should be

equal:

$$\begin{aligned} R_{\text{exp}} \cdot c_{ss} \\ &= \left[\left(s - \left(\langle \vec{d}^* \vec{d}\rangle^{-1} \cdot \langle \vec{d}^* s\rangle\right) \cdot \vec{d}\right)^* \left(s - \left(\langle \vec{d}^* \vec{d}\rangle^{-1} \cdot \langle \vec{d}^* s\rangle\right) \cdot \vec{d}\right)\right] \\ &= [s^* s] - \left(\langle \vec{d}\vec{d}^*\rangle^{-1} \cdot \langle \vec{d} s^*\rangle\right) \cdot \left[\vec{d}^* s\right] - \left(\langle \vec{d}^* \vec{d}\rangle^{-1} \cdot \langle \vec{d} s^*\rangle\right) \cdot \left[\vec{d} s^*\right] \\ &\quad + \left(\langle \vec{d}\vec{d}^*\rangle^{-1} \cdot \langle \vec{d} s^*\rangle\right) \cdot \left[\vec{d}^* \vec{d}\right] \cdot \left(\langle \vec{d}^* \vec{d}\rangle^{-1} \cdot \langle \vec{d} s^*\rangle\right) \\ &= R_{\text{theo}} \cdot c_{ss} \end{aligned} \tag{17}$$

if averages over the training and the test set are equivalent.

$$p = \frac{\langle \xi_P^* \xi_P \rangle}{\langle \xi_P^* \xi_P \rangle + \langle \xi_S^* \xi_S \rangle} = \frac{1}{1 + \frac{\langle A_S^2 \rangle}{\langle A_P^2 \rangle}} = \frac{1}{1 + \frac{\frac{N_S}{N}\langle A^2 \rangle}{\frac{N_P}{N}\langle A^2 \rangle}} = \frac{N_P}{N} \equiv r_P \tag{18}$$

$$\delta\phi_{\text{surf}} = -G\rho_0\xi_0 \int_0^{2\pi} \mathrm{d}\varphi \int_0^{\pi} \mathrm{d}\vartheta \, a^2 \sin\vartheta \frac{\vec{e}_r \cdot \vec{e}_{P,S}}{a} \mathrm{e}^{i(ka\vec{e}_r \cdot \vec{e}_{P,S_\perp} - \omega t)} \tag{19}$$

choose $\vec{e}_{P,S} = \vec{e_z}$ in the limit where $ka \ll 1$:

$$\delta\phi_{\text{surf}} = -2\pi G\rho_0\xi_0 a \mathrm{e}^{-i\omega t} \int_0^{\pi} \mathrm{d}\vartheta \, \cos\vartheta = 0 \tag{20}$$

$$\begin{aligned} \delta\phi_{\text{surf}} &= -2\pi G\rho_0\xi_0 a \mathrm{e}^{-i\omega t} \int_0^{\pi} \mathrm{d}\vartheta \, \sin(\vartheta)\cos(\vartheta)(1 + \mathrm{i}ak\cos(\vartheta))(1 - \frac{r_0}{a}(\cos(\vartheta)\cos(\vartheta_0))) \\ &= \frac{4\pi}{3} G\rho_0\xi_0 \mathrm{e}^{-i\omega t}(r_0 \cos(\vartheta_0) - \mathrm{i}a^2 k) \end{aligned} \tag{21}$$

$$\delta\vec{a} = -\vec{\nabla}\delta\phi = -\frac{4\pi}{3} G\rho_0\xi_0 \mathrm{e}^{-i\omega t}(\cos(\vartheta_0)\vec{e}_{r_0} - \sin(\vartheta_0)\vec{e}_{\vartheta_0}) = -\frac{4\pi}{3} G\rho_0 \vec{\xi}(0,t) \tag{22}$$

References