Space Py Quest 1.0

Isobel Romero-Shaw, Roshni Vincent and Andreas Freise

with support from Phillip Jones

February 27, 2018

Space Py Quest is a gravitational-wave detector game based on Space Time Quest [1], providing on open source implementation of the same concept for educational purposes. The Space Py Quest source code is hosted at https://github.com/gwoptics/SpacePyQuest. This document is also available as LIGO document number T1800061.

1 Introduction

The game Space Time Quest has been developed by gravitational-wave group in Birmingham for outreach and public engament with science [2]. The game puts you in charge of designing your own gravitational wave detector. You make choices and trade-off decisions to select the best technology for your detector while keeping an eye on the budget. The game is available as a modern application ('app') on the app stores for iOS, Android, and various PC system, and provides an online leaderboard, see https://www.laserlabs.org.

Space Py Quest aims to enhance the educational capacities of Space Time Quest by providing users with the opportunity to examine and alter the source code of the game. The two games have similar underlying mechanics, but the methods by which a user interacts with each encourage different learning outcomes. Both games compute the internal noise of a gravitational-wave detector using identical equations. Space Py Quest introduces some additional, arbitrary scaling factors that prevent the games giving identical results. Without these changes, players could use Space Py Quest and numerical optimisation methods to obtain the top score in the online leaderboards of Space Time Quest's high scores.

Space Py Quest presents an open-source interface to the game logic through a Jupyter notebook [3] with an auto-updating plot that adjusts according to the detector parameters set using widgets such as sliders [4]. It has been written in Python because this high-level language is widely familiar and accessible, allowing other students to read, understand and change the code. Python is already being taught in primary schools, making Space Py Quest an ideal teaching tool for the future.

Both Space Quest games are based on the scientific modelling software used in the gravitational-wave community for the design of new detectors. Figure 1 shows the sensitivity of the LIGO detectors calculated by GWINC, a scientific software package developed in the gravitational-wave community. As a player, your task is to lower the detector noise while studying a similar plot.

The games are fun to play but are meant to also be used in an educational context. After playing the game you will

- know about the different systems that are part of a gravitational-wave detector,
- understand how the internal noise determines the sensitivity of the detector
- be able to run a simple Jupyter notebook running Python code
- have experience in performing inevitable trade-off decisions between different enhancements, to keep the detector costs within a set budget.

This document fills the gap between both Space Quest games and actual sensitivity modelling software used in the scientific gravitational-wave collaborations, by providing the documentation for the noise calculations used in the games. Space Py Quest's noise curves have the same scaling and overall shape of their real counterparts, but the functions that it uses to obtain them are accessible to the user through the code, and are referenced and explained within this text. Space Py Quest should be simple enough, so that it can be successfully understood and played within a timeframe short enough to entertain, but not bore, its player. Thus, although the functions used to generate noise curves are *based* on physically correct equations, they are simplified to avoid over-complicating the game. An average user, for example, will benefit from learning that less excess gas leads to more detections, and may be interested to know that this is to do with the gas particles interrupting the laser beam; they are less likely to want to know exactly how the velocity of each particle affects the residual gas noise curve.

This document is structured as follows: Section 2 provides the equations used by the game to calculate the noise curves and the resulting range of the detector. Section 4 describes the game design and code. Section 3 details the testing of Space Py Quest, achieved by ensuring that the noise curves and parameters behave as dictated by the noise equations and realistic expectations. Ideas for future development of the game are given in the discussion in section 5.1. Any utility functions used by the code and all constants referenced in equations presented in this document are provided in the appendix.
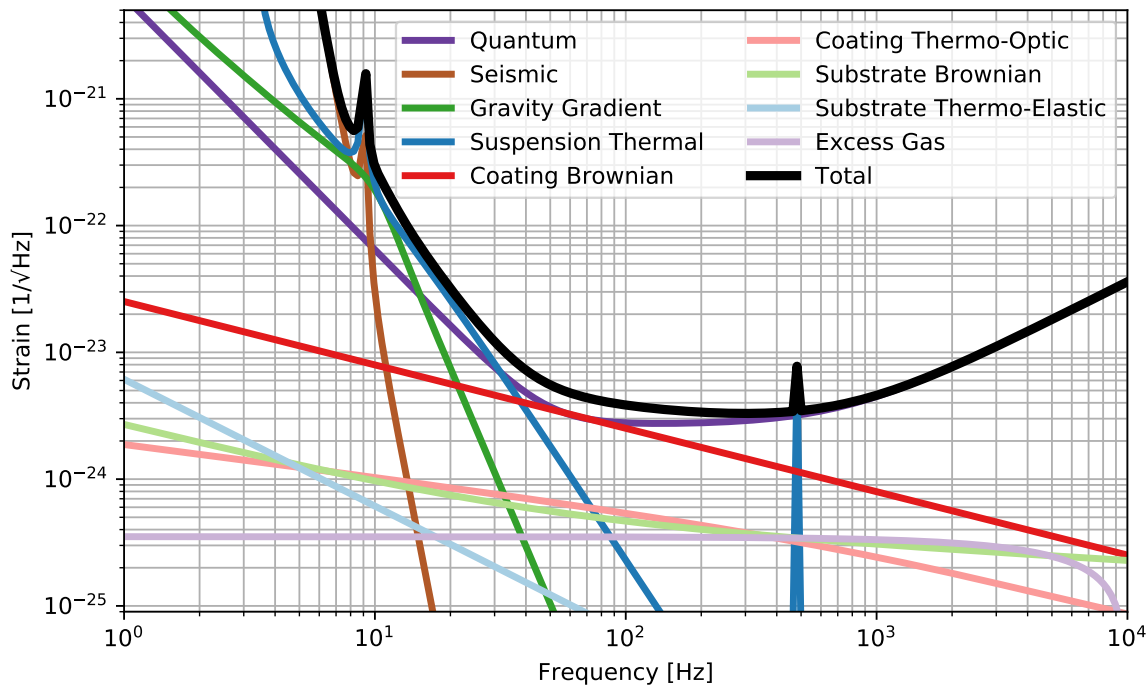
Figure 1: The central piece of the Space Quest games is a plot that shows the noise budget and sensitivity of the instrument. This plot is the equivalent plot produced by the scientific software GWINC used in the LIGO collaboration. Each coloured trace shows a different detector noise and the black traces, which is the quadratic sum of the all other traces, presents the sensitivity of the detector. The y-axis is a 'strain' spectral density, with 'strain' being the amplitude of gravitational waves, and the x-axis is the frequency of the gravitational wave signal. When the black curve is lower then the detector can detect gravitational waves with a smaller amplitude. The aim of the quest games is to achieve lower numerical values for this black trace, thus improving the sensitivity of the detector.

1.1 Space Time Quest and Space Py Quest

Space Time Quest is designed to be played by a wide range of mostly non-experts, who are interested in gravitational waves and are motivated to keep playing to get the best score, beating their friends or winning a high place on the leaderboard. Space Py Quest can also be played by non-experts, but is more apt for those who wish to interact with the driving code or understand the noise curves. Some differences between the two games are listed and discussed below.

1. **Format**
   Space Time Quest is an app with eye-catching graphics and clear transitions between stages of the game. Space Py Quest exists within a Jupyter notebook, which is not as graphically interesting. It requires some interaction with code, with a user handling a set of parameters in order to initialise the detector as either aLIGO or LIGO Voyager. This introduces Python's *dictionary* type to the player, and demonstrates how Space Py Quest can provide programming education in addition to the lessons taught by Space Time Quest. It is not yet as visually appealing, but this could also be something for a user to alter themselves. The motivation then slightly deviates from that of Space Time Quest - *Build Your Own Detector Sensitivity Modelling Software* rather than *Build Your Own Detector*.

2. **Narrative**
   In Space Time Quest, the game follows the user as they set up their detector in a certain site, lower the sensitivity curve as far as possible within the budget, and run the detector to find their score. As such,

there is a clear beginning, middle and end to each attempt. Space Py Quest was designed for a more flexible interaction. The user may change the site of the detector at any point, and may run the detector to find its range as many times as they like without having to restart the game. Again, this is more in line with a *Make Your Own Detector Sensitivity Modelling Software* aim - whilst detectors themselves might not be easy to change once built, detector designs are flexible and cost-effective ways of trialing new ideas.

3. **Removable Noise Curves**
   Users of Space Py Quest are able to choose which noise curves are displayed using tick-box widgets, allowing the effects of different parameters on individual noise sources to be investigated.

4. **Integration Methods**
   Space Time Quest was originally written in Java and later implemented in C#, as required by the respective game engines. Space Py Quest is written in Python and makes use of the SciPy library functions for fast numerical computations, for example Simpson's integration function. The use a different integration methods and numerical round-off errors lead to slight differences in the scores between Space Time and Space Py Quest.

2 Physics and Math Behind the Game

2.1 Noise Curves

Noise curves displayed by Space Py Quest are scattered throughout this document. Figure 2 is an example of the plots generated by the game, with additional dashed traces highlighting the basic slopes and shapes of each individual noise.

Gravitational waves stretch and squash the long arms of the detector, which can be expressed as a fractional length change, or strain. Strain equivalent *noise* arises when other effects that are not gravitational waves cause the detector signal to change. In Space Py Quest, strain equivalent noise is illustrated as an amplitude spectral density in the frequency domain, allowing the user to view a time-averaged snapshot of the strain that the noise sources emulate.

As displayed in figure 2, the *total* sensitivity curve is the vector sum of the individual noise contributions. This is the limit of the detector's sensitivity, and a lower total noise means a better sensitivity.

The following sections give a technical description of the simplified individual noise curves.

Ground Motion Noise

- **Seismic**
  The base seismic noise is

$$X_0 = \frac{X_{\text{dc}}}{1 + \left(\frac{f}{f_c}\right)^{n_0}} + X_{\text{hf}}, \tag{1}$$

  where X_{dc}, X_{hf}, n_0 and f_c are location-specific parameters that are detailed in appendix A.1. X_0 experiences a reduction if the detector is located a distance d underground, so that

$$\mathcal{R} = \frac{1}{\sqrt{1 + \left(\frac{d}{50}\right)^4}} + 0.8 \times 10^{-3}. \tag{2}$$

  These two terms combine to give the final displacement due to seismic activity,

$$X_{\text{seis}} = X_0 \times \mathcal{R}. \tag{3}$$

  The seismic noise is calculated considering a highly-damped pendulum with $1 < Q_{\text{pend}} < 10$, where Q_{pend} is the pendulum quality factor. Here, we assume a quality factor of $Q_{\text{pend}} = 5$. The ground motion, X_{seis},
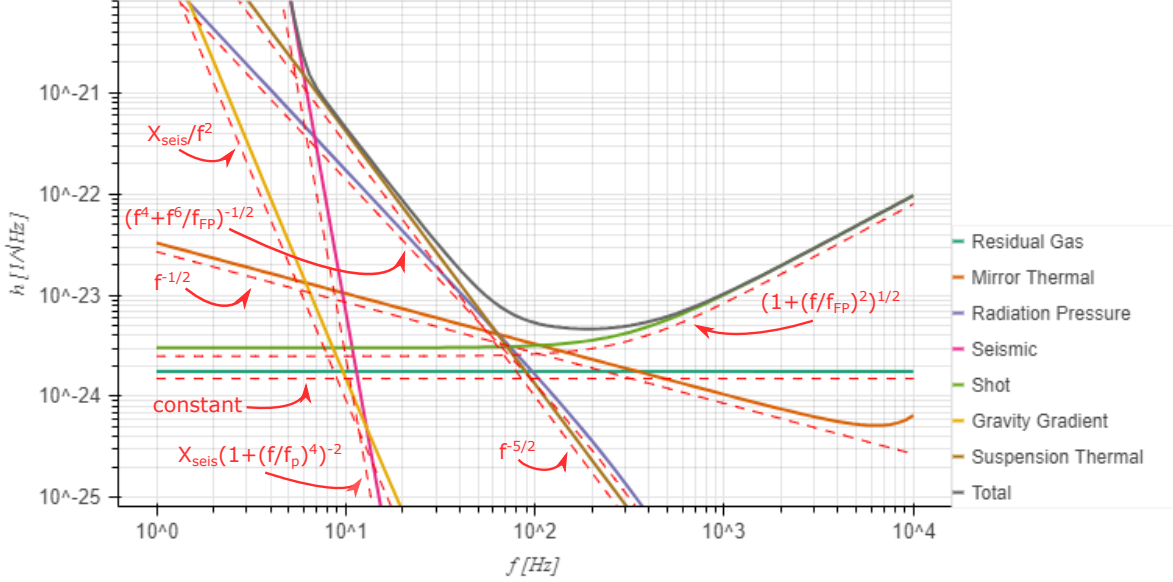
Figure 2: The default aLIGO noise curve model generated by Space Py Quest. The solid traces represent the graphical output of Space Py Quest, the dashed traces have been added here to indicate the basic slope or shape of the respective noise curves described in this section. The detector used in this example is initialised at a depth, d, of 0 m, in the desert site, with fused silica mirrors. The mirrors have a roughness, R, of 1 nm, and a mass, M, of 40 kg. There are 4 suspension stages, N_s, and the suspension has a length, l, of 60 cm. 6 vacuum pumps, N_p, are used and the detector temperature, T, is 295 K. The laser power, P, is set to 125 W. Unless specified otherwise, plots in this section will refer to a detector with this configuration.

is calculated as in equation 3. The pendulum oscillation frequency, f_p is given as

$$f_p = \frac{1}{2\pi}\sqrt{\frac{g}{l}}, \tag{4}$$

where g is the gravitational acceleration and l is the suspension length. We consider only $f > f_p$, since for the shortest pendulum we get an f_p of ~ 1 Hz. The transfer function between ground motion and the test mass movement is then calculated as

$$\mathcal{T}_{\text{pend}} = \frac{1}{1 + \left(\frac{f}{f_p}\right)^4 - \left(2 - \frac{1}{Q_{\text{pend}}}\right)\left(\frac{f}{f_p}\right)^2}. \tag{5}$$

The seismic noise is then

$$h_S = \frac{2}{L}X_{\text{seis}}\left(\sqrt{\mathcal{T}_{\text{pend}}}\right)^{N_s}. \tag{6}$$

- **Gravity Gradient (Newtonian)**
  The Newtonian, or gravity gradient, strain noise arises from modulation to the local gravitational field due to density perturbations of the Earth [5]. This calculation assumes the form of the expressions for gravity gradient noise given in equations (5) and (6) of the 2004 Virgo Sensitivity Curve document [6]. The Newtonian noise in the 2016 version of Space Time Quest is based on equation (6),

$$h_{\text{GG}} = \frac{X_{\text{seis}}(1.3 \times 10^{-8})}{Lf^2}, \tag{7}$$

where

$$(1.3 \times 10^{-8}) \approx \frac{2.7G\rho_E\sqrt{2}}{(2\pi)^2}. \tag{8}$$

5

G is Newton's constant and ρ_E is the density of the Earth, $\sim 2 \times 10^3$ kgm^{-3}. Equation (5) from the Virgo sensitivity document returns a strain amplitude about 7 times larger than equation 7 here, the calculation used in Space Py Quest.
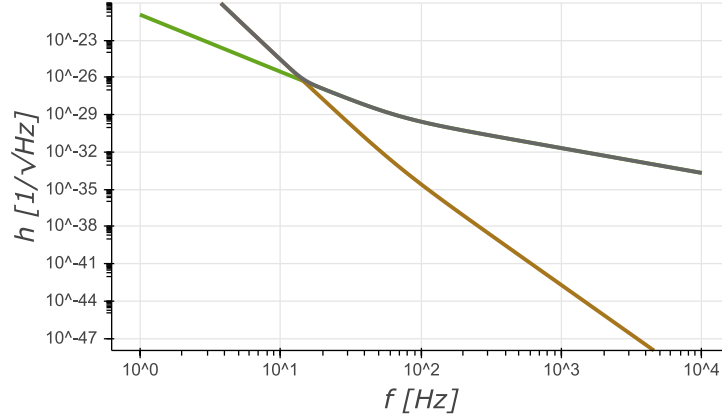

Figure 3: Approximate ground noise models for the aLIGO detector described in figure 2. Seismic oscillations affect the detector over the entire frequency spectrum, but at mid-frequencies and above its contribution is extremely small in comparison to strain noise from other sources.
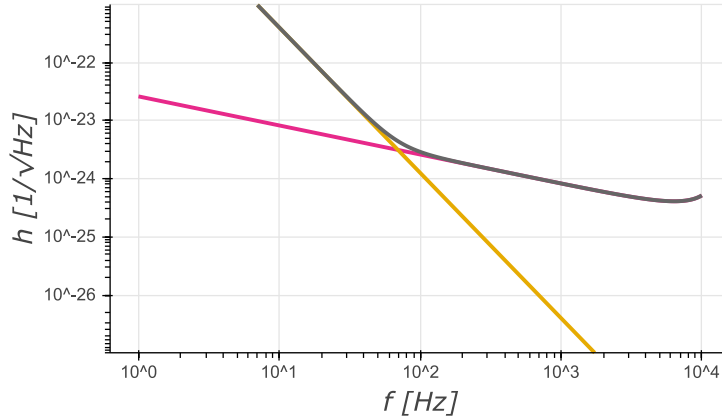
Thermal Noise

Figure 4: The thermal noise models used by Space-Py Quest, which contribute significantly at the lower end of the frequency range.

- **Mirror Thermal**
  The correct power spectral density of the mirror's internal vibrations, as provided in [6], is

$$X_{\text{int}}^2 = \frac{4 k_b T}{\omega^2} \times \mathbf{Re}\left(\frac{1}{Imp(\omega)}\right). \tag{9}$$

Here, k_b is the Boltzmann constant, $Imp(\omega)$ is the system's mechanical impedence, and $\omega = 2\pi f$. In order to calculate this for Space Py Quest, we approximate only equation (28) from [6], which is used to calculate the fluctuations due to the first mode resonance. The expression

$$w_1 = \times A_{\text{MT}} \times \left(\frac{23}{M}\right)^{\frac{2}{3}}, \tag{10}$$

6

has been chosen to represent the first resonant frequency of the mirror, so that it scales inversely with mirror mass. This is because the Space Py Quest mirrors are assumed to maintain their density and aspect ratios, so higher mass means larger reflective surface area. A_{MT} is a scaling factor, here set to 20505, and M is the mirror mass. This gives ω_1 about 4 times larger than those given in [6] for masses of ~ 20 kg, but only about twice as large for masses of ~ 50 kg.

We use the 'effective' mirror mass associated with the w_1 mode, M_{eff},

$$M_{\mathrm{eff}} = 0.28 \times M. \tag{11}$$

This reduction follows the effective masses of ~ 6.5 given in [6] for masses of ~ 20 kg.

The Space Py Quest model returns a strain noise amplitude of

$$h_{\mathrm{MT}} = \frac{4}{L} \sqrt{\frac{k_b T}{M_{\mathrm{eff}} \mathcal{Q}}} \times \frac{w_1}{\sqrt{w\left((w_1^2 - w^2)^2 + \left(\frac{w_1^2}{\mathcal{Q}}\right)^2\right)}}. \tag{12}$$

This incorporates our approximation of equation (28) from [6] for the first-mode fluctuations into the relevant strain calculation given in equation (29).

- **Suspension Thermal**
  In reality, the suspension wires contribute *pendulum* (horizontal) oscilliations, *vertical* oscillations, and violin modes [7] to the thermal noise. As in equations (19), (22) and (24) in [6], each contribution has the form

  $$h_{\mathrm{ST}} = \frac{2}{L} \sqrt{X_{\mathrm{ST}}^2}, \tag{13}$$

  where X_{ST} are the thermal fluctuations in the suspension material and L is the detector's arm length. We initially make the simplifying assumption that all suspension stages are the same, and that the last stage is the only one contributing to the thermal noise. We also assume that there are 4 suspension stages and that the wires are made of steel, as in the Virgo design document [6]. Finally, the violin modes are not included for simplification. Thus, the Space Py Quest calculation uses Young's Modulus, $E \approx 2 \times 10^{11}$ Pa, a breaking strength of $Y \approx 2 \times 10^9$ Pa, and a loss angle of $\phi \approx 1 \times 10^{-4}$. The suspension thermal fluctuations are expressed as

  $$X_{\mathrm{ST}}^2 = \frac{4 k_b T}{\omega^5} \frac{g}{4l^2} \sqrt{\frac{gE}{\pi M} \frac{\phi}{Y}}, \tag{14}$$

  which can be substituted into equation 13 to return

  $$h_{\mathrm{ST}} = \frac{2}{lL} \times \left(\frac{gE}{\pi M}\right)^{\frac{1}{4}} \times \sqrt{\frac{k_b T g \phi}{\omega^5 Y}}, \tag{15}$$

  where l is the suspension length, T is the detector's temperature, M is the mirror mass, and $\omega = 2\pi f$ as before.

Residual Gas

The residual gas pressure influences the refraction index of the inner interferometer arms,

$$n = 1 + \epsilon \frac{\mathcal{P}_{\mathrm{arm}}}{\mathcal{P}}, \tag{16}$$

where $\epsilon \approx 1.2 \times 106{-}4$ [6], and $\mathcal{P}$ is the atmospheric pressure in mbar. The Virgo design document gives the strain due to radiation pressure fluctuation as

$$h_{\mathrm{RG}} = \frac{\epsilon \pi^{\frac{1}{4}}}{N_{\mathrm{atmos}}} \sqrt{\frac{w_{\mathrm{beam}} N_{\mathrm{arm}}}{v_{\mathrm{H2}} V_{\mathrm{beam}}}} \approx 2.5 \times 10^{-26}, \tag{17}$$
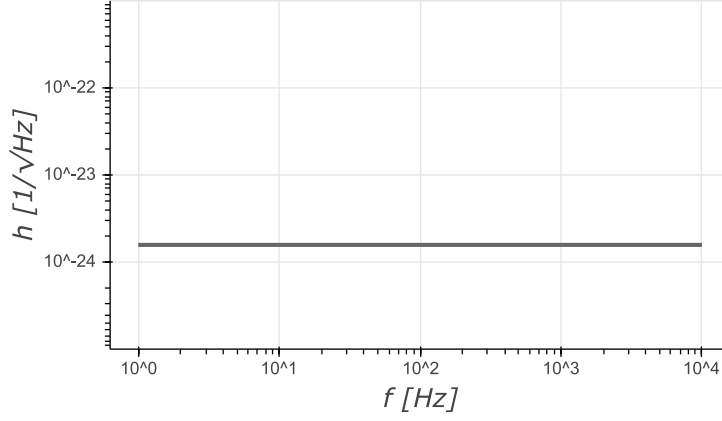
Figure 5: The residual gas noise model is a limiting 'floor' to the noise model. It is not frequency-dependent, so appears as a straight line across the graph.

where w_{beam} is the beam waist far from the mirror ($\sim 0.1m$), v_{H2} is the molecular velocity, $V_{\text{beam}} = \pi w_{\text{beam}}^2 L_{\text{arm}}$ is the 'beam volume', and N_{arm} and N_{atmos} are the molecular densities in the arm and in the atmosphere, respectively. The arm pressure, P_{arm}, is calculated in units of mbar as

$$\mathcal{P}_{\text{arm}} = \mathcal{P}e^{-8N_p} + 10e^{-4N_p} + 10^{-3}e^{-2N_p} + 10^{-8-0.7N_p} + 10^{-11}e^{-0.3N_p} + 10^{-16}. \tag{18}$$

This is an approximation of the sum of partial pressures of the main gasses contributing to the residual gas. The final component of this calculation is the only part that is independent of the number of vacuum pumps, N_p. It is thus the minimum pressure that a user can reach. The equation used to calculate the strain due to residual gas pressure fluctuations is then

$$h_{\text{RG}} = 1.37 \times 10^{-18} \times \sqrt{\frac{\mathcal{P}_{\text{arm}}}{L}}. \tag{19}$$
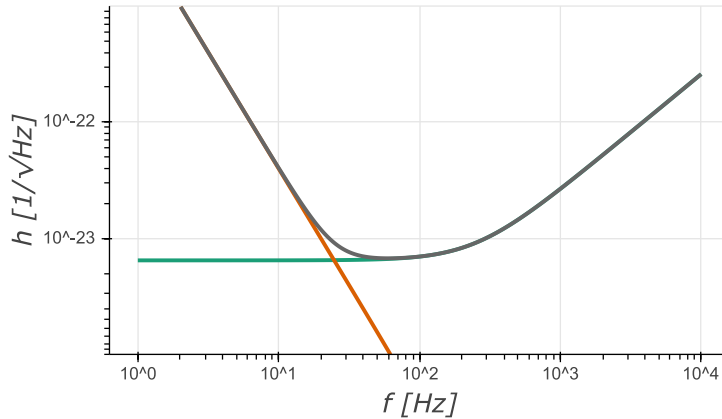
Quantum Noise

Figure 6: The quantum limit in Space-Py Quest, formed by the combination of radiation pressure noise and shot noise.

- **Radiation Pressure**

8

The radiation pressure noise calculation in Space Py Quest is an adaption of equation (12) in [6],

$$h_{\mathrm{RP}} = \frac{16\sqrt{2}\mathcal{F}}{LM(2\pi f)^2} \sqrt{\frac{hF_{\mathrm{PR}}P}{4\pi^2 c\lambda}} \sqrt{\frac{1}{1 + \left(\frac{f}{f_{\mathrm{FP}}}\right)^2}}. \tag{20}$$

In this expression, $\mathcal{F}$ is the arm cavity finesse, F_{PR} is the power recycling factor, c is the speed of light, λ is the wavelength of the laser light, and f_{FP} is the Fabry-Perot cut-off frequency, defined as

$$f_{\mathrm{FP}} = \frac{c}{4L\mathcal{F}}. \tag{21}$$

We multiply the result by the square root of the product of the mirror material damping rate $\mathcal{L}$ and the mirror surface roughness loss $\mathcal{L}_R$, $\sqrt{\mathcal{L}\mathcal{L}_R}$. The power scaling of these values is again arbitrary. The radiation pressure noise is then

$$h_{\mathrm{RP}} = \frac{\mathcal{F}}{LM\pi^3 f^2} \sqrt{\frac{8hF_{\mathrm{PR}}P}{c\lambda}} \sqrt{\frac{1}{1 + \left(\frac{f}{f_{\mathrm{FP}}}\right)^2}} \times \sqrt{\mathcal{L}\mathcal{L}_R}. \tag{22}$$

- **Shot**
  Shot noise is given in equation (11) of [6] as

$$h_{\mathrm{Sh}} = \frac{1}{8L\mathcal{F}} \times \sqrt{2h\frac{\lambda c}{\eta F_{\mathrm{PR}}P}} \times \sqrt{1 + \left(\frac{f}{f_{\mathrm{FP}}}\right)^2}. \tag{23}$$

In Space Py Quest, we assume that a photodetector efficiency of $\eta = 1$. We also include the lossiness due to the roughness of the surface, so that the noise calculation used is

$$h_{\mathrm{Sh}} = \frac{1}{8L\mathcal{F}} \times \sqrt{2h\frac{\lambda c}{F_{\mathrm{PR}}P}} \times \sqrt{1 + \left(\frac{f}{f_{\mathrm{FP}}}\right)^2} \times \mathcal{L}^{-\frac{1}{2}}\mathcal{L}_R^{-5}. \tag{24}$$

The powers to which $\mathcal{L}$ and $\mathcal{L}_R$ are raised are arbitrary.

2.2 Detector Score Calculations

Detector Range

In Space Py Quest, the detector range is the distance to which the detector can observe mergers of two given masses, m_1 and m_2, based only on the inspiral section of their signal. During this stage of the coalescence, the signal amplitude spectral density in the frequency domain has roughly the same gradient as the total noise amplitude spectral density, and goes as $f^{-7/3}$ [8]. The merge and ringdown sections of the signal, which occur at higher frequencies, are not considered in this calculation. This has the effect of rendering the high-frequency sensitivity of the Space Py Quest detector irrelevant to the detector range for heavier masses, whose inspiral signal terminates at relatively low frequencies within the detector's frequency band.

The net noise power spectrum, $\mathcal{S}$, is the sum of the squares of all noises at each point in the detector's sensitive frequency band. The *sensitivity integral* is the integral of the signal-to-noise ratio,

$$\mathbf{I}_S(f) = \int_{f_{lo}}^{f_{hi}} \frac{f^{-\frac{7}{3}}df}{\mathcal{S}}, \tag{25}$$

where f_{lo} and f_{hi} denote the lower and upper limits of the frequency range, respectively. The Keplerian frequency at the innermost circular orbit of a binary inspiral is

$$f_{\mathrm{isco}} = \frac{1}{6^{1.5}\pi(m_1 + m_2)M_{\odot,\mathrm{nu}}}, \tag{26}$$

Where $M_{\odot,\text{nu}} = \frac{M_{\odot}G}{c^3}$ is the Sun's mass in natural units. At f_{isco}, which point the gravitational wave inspiral signal stops for lower masses, and transitions into the actual merge and then ringdown signals for more massive binaries. This becomes the upper limit of the sensitivity line integral, $\mathbf{I}_S(f)$.

The combined mass of the binary is defined by its chirp mass,

$$\mu = \frac{(m_1 m_2)^{0.6}}{(m_1 + m_2)^{0.2}} M_{\odot,\text{nu}}. \tag{27}$$

The detector distance is then calculated by

$$\mathcal{D} = \frac{\sqrt{\mathbf{I}_S(f) \times \mathcal{M}}}{2.26 \times Mpc_{\text{nu}}}, \tag{28}$$

where

$$\mathcal{M} = \frac{80\mu^{\frac{5}{3}}}{96\pi^{\frac{4}{3}}\tau_{\text{snr}}^2}. \tag{29}$$

and $\tau_{\text{snr}} = 8$ is the signal-to-noise ratio required for observation. $Mpc_{\text{nu}} = \frac{Mpc}{c}$ is Mpc in natural units. The equation for $\mathcal{D}$ is used to find the ranges to which the detector can observe binary black hole mergers, r_{bhbh}, and binary neutron star mergers, r_{nsns}. For the black holes, both masses are taken as $47\,M_{\odot}$, whilst for neutron stars, the masses are both $1.7\,M_{\odot}$.

Detector Cost

The total cost, $\mathcal{C}$, is the sum of parameter-dependent costs, C_i. Its dependence on the number of suspension stages N_s, suspension length l, and mirror mass M is contained within the component that considers vibration, C_{vib}. There is also a dependence on mirror mass in the calculation of the roughness cost, C_r, which is influenced additionally by the mirror roughness R and the roughness losses $\mathcal{L}_R$. The laser power, P, and detector depth, d, contribute costs C_{pow} and C_{depth} respectively. The cooling cost, C_{temp}, is dependent on the detector temperature T, its initial ambient temperature T_0, the temperate change, ΔT, per kilometer, and the temperature of nitrogen, T_N. These costs are calculated as given below.

$$C_{\text{depth}} = \begin{cases} (d-20)^{\frac{1}{3}} \times 75 \times 10^5 & \text{when } (d-20) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{30}$$

$$C_{\text{temp}} = \begin{cases} A_{T,1} \times \left(T_0 + \frac{d\Delta T}{100} - T\right) & \text{when } T > T_N \\ 7 \times 10^6 + A_{T,2}\left(77 - T\right)^2 & \text{otherwise} \end{cases} \tag{31}$$

(where arbitrary constants $A_{T,1}$ and $A_{T,2}$ are 20102 and 10201, respectively) (32)

$$C_{\text{Np}} = N_p \times C_v \tag{33}$$

(where P_{hi} is the upper power limit) (37)

$$C_r = \frac{M^{\frac{2}{3}}(R^3 - \mathcal{L}_R^3)(8 \times 10^7)}{25} \tag{34}$$

$$C_{\text{vib}} = l^{a_v} \times N_s^{b_v} \times M^{c_v} \times 60 - 60 \tag{38}$$

$$C_{\text{mat}} = \mathcal{C}_{\text{mat}} \tag{35}$$

(where a_v, b_v and c_v are 2.1, 5.5 and 1.2 in that (39)

$$C_{\text{pow}} = 47 \times 10^3 + 25 \times 10^6 \times \left(\frac{P}{P_{\text{hi}}}\right)^2 \tag{36}$$

order, and a_v and c_v are arbitrary) (40)

Then

$$\mathcal{C} = \sum_i C_i, \tag{41}$$

where i represents a parameter with a unique cost.

Numbers of Detections (and Missed Detections)

- **Binary Neutron Star Mergers**
  The number of black hole binary merge signals detected is the rounded result of

$$N_{\text{nsns}} = \frac{4\pi \times 6000}{3 \times 12} \left(\frac{r_{\text{nsns}}}{1 \times 10^3} \right)^3, \tag{42}$$

  where r_{nsns} is the range to which the detector can sense neutron star mergers with component masses of 1.7 $M_\odot$.

- **Binary Black Hole Mergers**
  The number of neutron star binary merge signals detected is the rounded result of

$$N_{\text{bhbh}} = \frac{4\pi \times 20}{3 \times 12} \left(\frac{r_{\text{bhbh}}}{1 \times 10^3} \right)^3, \tag{43}$$

  where r_{bhbh} is the range to which the detector can sense black hole mergers with component masses of 47 $M_\odot$.

- **Supernovae**
  Detectors with good mid- to high-frequency sensitivity can obtain just *one* supernova detection. This was done after the tests described in section 3 were performed. The supernova signal amplitude curve is found using the total noise amplitude curve $\mathcal{S}$,

$$y_s = \sqrt{\mathcal{S} + (1 \times 10^{-23})^2}. \tag{44}$$

  The area under both the noise and the supernova curves is calculated by taking their integrals, $\mathcal{I}_n$ and $\mathcal{I}_s$ respectively. The result is then

$$N_{\text{sn}} = \begin{cases} 1 & \text{if } 25 \times \left(\frac{\mathcal{I}_s - \mathcal{I}_n}{4 \times 10^{-20}} \right)^3 \geq 1 \\ 0 & \text{otherwise.} \end{cases} \tag{45}$$

- **Complexity**
  The 'complexity', $\mathcal{Z}$, is a measure of how challenging it is to maintain the detector. If a detector requires a lot of time to restore it to its design sensitivity when it breaks, then it spends less time observing, and events can be missed as a result. The **total** complexity is the **sum** of complexities due to the detector's depth d, and temperature T, the number of vacuum pumps N_p and suspension stages N_s, the laser power P, and the mass M, material damping rate $\mathcal{L}$, and roughness R of the mirrors. The complexities are:

$$Z_{\text{temp}} = \begin{cases} 1 - \frac{T - T_N}{T_0 + \frac{d\Delta T}{100} - T_N} & \text{when } T > T_N \\ 5 & \text{otherwise} \end{cases} \tag{46}$$

$$Z_r = 1 + \frac{50 - R}{500} \tag{47} \qquad\qquad Z_{\text{Ns}} = \frac{N_s}{2} \tag{49}$$

$$Z_{\text{Np}} = \frac{N_p}{10} \tag{48} \qquad\qquad Z_{\text{pow}} = \frac{P}{10} \tag{50}$$

$$Z_{\text{mass}} = \frac{M}{50} \qquad (51)$$

$$Z_{\text{mat}} = \mathcal{L} \qquad (52)$$

Then

$$\mathcal{Z} = Z_{\text{depth}} + \sum_i Z_i, \qquad (53)$$

where the complexity due to the detector's depth, Z_{depth}, is a linear interpolation of approximated depth and complexity data, and i represents a source of complexity.

- **Missed Detections**
  The number of missed events depends on the complexity of the detector, Z, itself in relation to the complexity credits of its site, Z_{cred}. The *overcomplexity*, $\mathcal{O}$, is

$$\mathcal{O} = \begin{cases} Z - Z_{\text{cred}} & \text{if } Z - Z_{\text{cred}} > 0 \\ 0 & \text{otherwise} \end{cases}. \qquad (54)$$

The complexity *scale*, S_Z, is then

$$S_Z = 1 - \frac{\mathcal{O}}{Z_{\text{cred}}}. \qquad (55)$$

The number of missed sources of a given type i is the rounded-down result of

$$N_{\text{missed,i}} = \begin{cases} N_i \times (1 - S_Z) & \text{if } N_i \times (1 - S_Z) > 0 \\ 0 & \text{otherwise}. \end{cases} \qquad (56)$$

3 Testing Space Py Quest

In order to ensure that Space Py Quest performs as expected, we completed a number of checks, including a comparison to the original Space Time Quest game, and a comparison to the actual sensitivity curves for aLIGO. We also checked tha the individual noise curves scaled as expected and that the maximum number of detections for different sources match the desired behaviour.

3.1 Testing Validity of Individual Noise Curves

The noise curves displayed by the Space Py Quest plotting interface were individually tested to ensure that they correspond accurately to their respective noise equations. This included examining the scaling of the curves with relevant. The scalings are considered in terms of the conventional log-log plot for detector sensitivity curves, and are outlined below. In addition, sensitivity values at arbitrary frequencies were evaluated, and can be seen in table 1.

As Space Py Quest is in essence a simplified gravitational wave detector sensitivity modelling software, the scaling of the noise curves can be compared to those produced by the more realistic *Gravitational-Wave Interferometer Noise Calculator* (GWINC) [9]. Parameters that approximately matched up to the aLIGO detector were input into Space Py Quest. These parameters were found using the aLIGO model used in GWINC, and consisted of $d = 0\text{m}$, $N_p = 6$, $N_s = 4$, $l = 0.6\text{m}$, $M = 40\text{kg}$, $P = 125\text{W}$, $r = 1\text{nm}$, $T = 295K$ and mirror material of Silica. The detector arm length, which is typically constant in the game, was set to 4000m to mirror aLIGO.

The aLIGO location is not an option is Space Py Quest, so the location that appeared to have the most similar values for seismic and gravity gradient noise (Island) was selected. Noise curves produced with these parameters and the aLIGO model in GWINC are shown in figure 7.

The quantum noise in GWINC includes both radiation pressure and shot noise, whereas the coating and substrate noises are combined into mirror thermal noise in Space Py Quest. As an overview, it can clearly be seen that total noise curves follows a similar shape, with minimum strain between $10^{-23}\text{Hz}^{-1/2}$ and $10^{-24}\text{Hz}^{-1/2}$. They
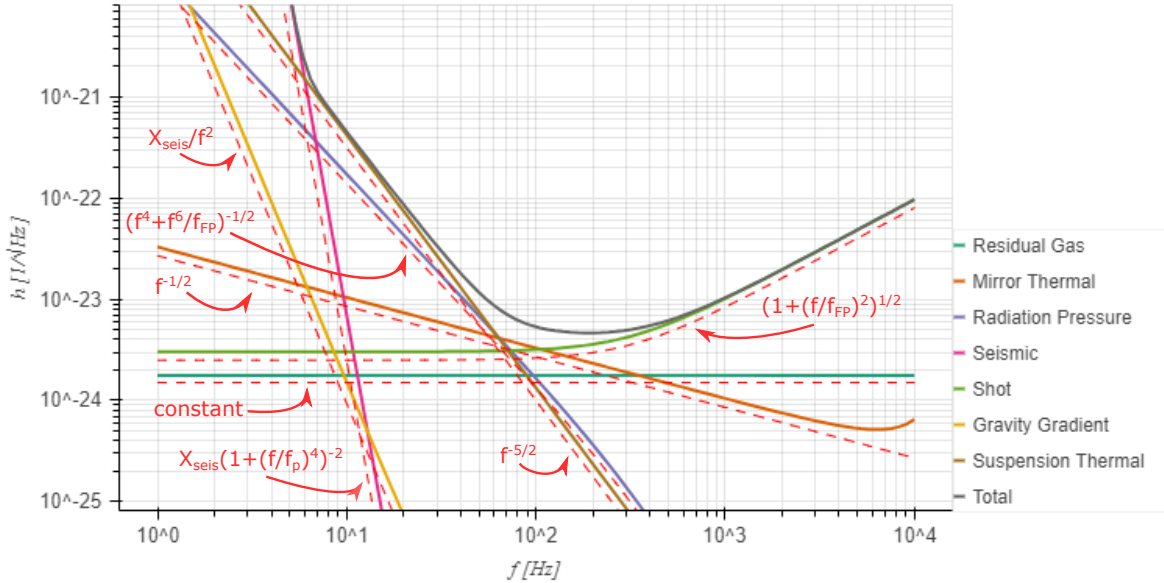
(a) Noise curves produced by Space Py Quest for aLIGO model
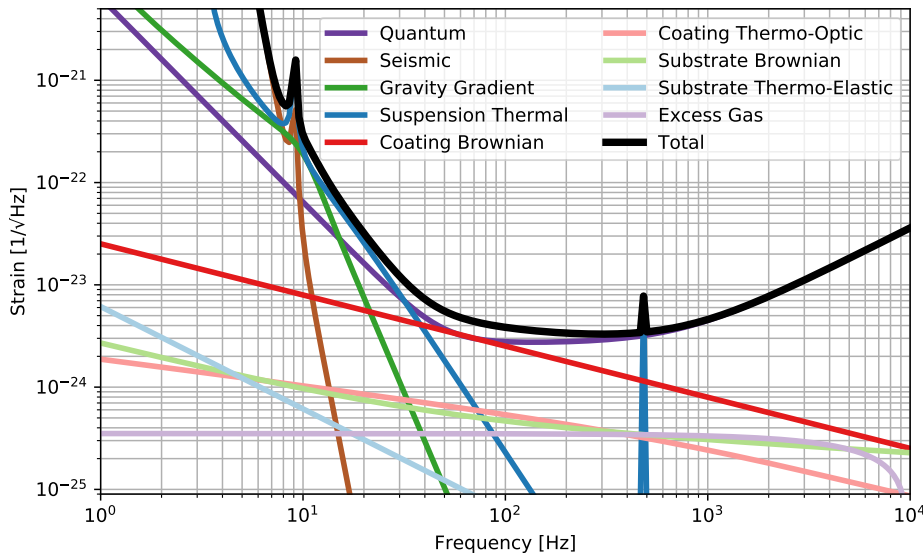
(b) Realistic noise curves produced by GWINC for aLIGO model

Figure 7: Noise curves for aLIGO model produced by Space Py Quest and GWINC. The maximum sensitivity for both appears to be around the same value, and both curves are mostly limited by seismic, suspension thermal and quantum shot noise.

both appear to be mostly limited by shot noise and suspension thermal noise, with seismic noise dominating at low frequencies. An obvious difference between the two is the greater number of spikes due to resonant modes included in GWINC. This difference is assumed to be due to the simplified equations used in the game. The scalings for individual noise curves are detailed below.

1. **Residual Gas Noise**

   - Frequency: Residual gas noise is the only noise that does not depend on frequency, and therefore is a horizontal line in the sensitivity plot at any parameter values. In the GWINC plot, however, this noise decreases at higher frequency, suggesting that a more realistic implementation of residual gas would have this effect.

13

- Number of pumps: The dependence on this parameter can be seen in equation 19. This equation suggests that although the addition of pumps lowers noise, this effect should decrease as N_p increases, which can be seen by altering this variable in Space Py Quest. The number of pumps was set to six to replicate the GWINC aLIGO model, but this appears to produce a higher value of noise. This is attributed to the simplification of residual gas used in the game.

2. **Mirror Thermal Noise**

- Frequency: This noise curve declines over frequency in the game, which can be explained by the $\omega^{-1/2}$ dependence in equation 12. At higher frequencies, the resonance in the $(\omega_1^2 - \omega^2)^{-1/2}$ term leads to a spike at the resonant frequency of the mirror. This can be seen more clearly in figure 8. The coating and substrate noises in GWINC are combined into the mirror thermal noise function, and it can be seen that the dominant coating Brownian noise follows a similar scaling and value to those produced by Space Py Quest, omitting the resonance. The resonance in the game is partially arbitrary.

- Mirror mass: The mass dependency in mirror thermal noise can be seen in the M_{eff} and ω_1 terms in equation 12. The first term suggests that increasing mass increases noise. However, the mass dependence in the second term means that as mass increases, noise increases, and appears to dominate. It also describes the resonant frequencies of the mirror, which decreases as mirror mass increases. This behaviour can be seen when varying the parameters in the game.

- Mirror material quality and temperature: The noise is proportional to $\sqrt{T}$, so as T increases the noise should increase. However, the optical loss dependence on temperature (described in appendix A.1) appears to dominate, meaning that after around 30K the noise curve decreases until approximately 250K, where it starts to increase in noise again.

3. **Radiation Pressure Noise**

- Frequency: This noise is proportional to $(f^4 + f^6/f_{\text{FP}})^{-1/2}$ as shown in equation 22, and the effect of the noise decreasing with frequency can be seen in Space Py Quest. In GWINC, radiation pressure is the dominant quantum noise at low frequencies, which appears to follow a similar scaling.

- Power and mirror mass: It is also proportional to $P^{1/2}$ and M^{-1}, and both responses can again be seen when altering the respective parameters in the game.

- Mirror material loss: Radiation pressure noise depends on the mirror material damping factor as $\mathcal{L}^{1/2}$. All material losses are identical except for Crystal, which is more lossy than the other materials and therefore has a smaller value of $\mathcal{L}$. This is depicted by a constant decrease in the radiation pressure noise across all frequencies when Crystal is selected in the game. Swapping between the other materials has no effect on this noise, as expected.

4. **Seismic Noise**

- Frequency, number of stages and location: The frequency dependence of seismic noise is encompassed in the $\mathcal{T}_{\text{pend}}$ and X_{seis} terms in equation 6, and is also dependent on the number of stages and location. Keeping the latter two constant, the $(f/f_p)^2$ value in $\mathcal{T}_{\text{pend}}$ term prevails at low frequency, meaning that noise is initially constant and then decreases with frequency. When the left and right hand sides of the $\mathcal{T}_{\text{pend}}$ denominator tend towards the same value, a resonance peak can be seen. At greater frequencies, the $(f/f_p)^4$ term dominates, meaning that noise again decreases with frequency. These frequencies dependencies are all multiplied by the $1 + (f/f_c)^{n_0}$ in X_{seis}, which too becomes less dominant at greater frequencies. All these effects can be seen in Space Py Quest, with the initial decrease in frequency being affected by changing location, and the resonance becoming more peaked at a greater number of stages. The number of stages is an exponent, so altering it should change the gradient of the line, which appears to be the case in the game. The seismic resonance can be seen at higher frequencies in GWINC, after which it follows a similar scaling to that in Space Py Quest.

- Suspension length: The suspension length dependency is similar to the frequency dependence in $\mathcal{T}_{\text{pend}}$, except with $f \to l^{1/2}$, which makes the distribution wider. Increasing the suspension length increases noise at very low frequencies but decreases it at values of 1Hz and higher, when the $X_{\text{seis}}(f/f_p)^4$ term is again dominant.

- Depth: The final parameter that affects seismic noise is depth. It is proportional to $(1 + \frac{d}{50}^4)^{-1/2}$, meaning that above 50m, noise decreases with depth. Similar to suspension length, the effect is smaller at larger depths. Below 50m, an increase in digging does not change the noise curve by a significant amount. This was confirmed by the Space Py Quest interactive plot.

5. **Shot Noise**

   - Frequency: The frequency dependence of this noise is $(1 + \left(\frac{f}{f_{\mathrm{FP}}}\right)^2)^{1/2}$, meaning that noise increases with frequency. This can be seen in Space Py Quest, where the noise appears constant until the frequency becomes the dominant term and the noise increases linearly with frequency. The shot noise is the dominant quantum noise at higher frequencies in GWINC, and again follows a similar scaling to the game.

   - Roughness: The roughness dependence is $(1 + (0.9/499)(1 - R))^{-5/2}$, meaning that as roughness increases, shot noise increases, with greater effect at greater roughness.

   - Power and mirror material loss: This noise is depends on power and mirror material damping factor according to $P^{-1/2}$ and $\mathcal{L}^{-1/2}$, meaning that an increase in either of these parameters decreases shot noise.

6. **Gravity Gradient**

   - Frequency: The frequency dependence of this noise is $X_{\mathrm{seis}}(f)/f^2$, and can be seen as a line of negative gradient in Space Py Quest. At higher frequencies, the $1/f^2$ becomes more dominant, and this transition can be seen as a 'knee' in the game. In GWINC, the more realistic gravity gradient noise leads to a different scaling at lower frequencies, until it transitions into a scaling that appears similar to Space Py Quest.

   - Location and depth: It is also dependent on location and depth, and this relation is identical to that in seismic noise. Therefore, the same effect can be seen when altering these parameters.

7. **Suspension Thermal**

   - Frequency: Suspension thermal has a frequency dependence $f^{-5/2}$, which is a power law and therefore is displayed as a straight line decreasing in value in Space Py Quest. Ignoring the realistic resonances included by GWINC, this noise curve again appears to follow a similar scaling to the game.

   - Mirror mass and suspension length: As the mass dependence is $M^{-1/4}$, the noise decreases with increasing mass, with the effect decreasing at greater mass. The noise also scales with suspension length as l^{-1}, which produces a similar effect to varying mass, but with a greater gradient.

   - Temperature: The suspension thermal noise is proportional to $\sqrt{T}$, so increases with temperature.

Check for varying detector configuations for each noise curves can be seen in table 1. All parameters are set to mirror the aLIGO model at the initiation of each check, and the original values were multiplied by the relevant scaling factors to ensure that they returned identical results to the noise calculations and the interactive plot in Space Py Quest.

3.2 Comparison to Space Time Quest

The original version of Space Py Quest ported to Python (before the arbitrary scaling factors were added) was compared to the released version of Space Time Quest. During this comparison, it was found that Space Time Quest did not include the suspension thermal noise when calculating the total noise and therefore the maximum detector range. Therefore, the suspension thermal noise was taken out of Space Py Quest when calculating the score, for a full comparison.

It was found that detector depth behaved differently in the two versions, despite the code being identical. Both games returned the same maximum number of detections, 281. The difference was very slight, meaning that with all parameters set to optimal values, the depth would need to be over 12m to achieve 281 detections in the C#

Table 1: Point checks for individual noise curves. The new strain obtained when altering selected parameters at varying frequencies is shown. The values obtained correspond to what can be seen in the Space Py when setting these parameters, and also what is output by multiplication by the relevant scaling factors, confirming how the plot and calculations are as expected.

Noise	Frequency [Hz]	aLIGO h [Hz$^{-1/2}$]	Altered Parameters	New h [Hz$^{-1/2}$]
Residual Gas	Any	1.8×10^{-24}	$N_p \to 16$	1.0×10^{-26}
Mirror Thermal	100	3.3×10^{-24}	$T \to 80$ K, $M \to 90$ Kg	7.3×10^{-23}
Radiation Pressure	1000	5.2×10^{-27}	$P \to 20$ W, $M \to 80$ Kg, Material $\to$ Crystal	6.5×10^{-28}
Seismic	5	1.0×10^{-20}	$l \to 3$ m, $N_s \to 5$, $d \to 800$ m	2.5×10^{-28}
Shot	100	3.2×10^{-24}	$R \to 100$ nm, $P \to 40$ W	1.5×10^{-23}
Gravity Gradient	10	1.5×10^{-24}	$d \to 500$ m, Location$\to$ City	6.8×10^{-25}
Suspension Thermal	100	1.3×10^{-24}	$M \to 20$ Kg, $l \to 3$ m, $T \to 70$ K	1.1×10^{-25}

version, whereas this was possible for any value from 0 to 20m in the Python version. This was concluded to be due to different rounding errors present in the two implementations.

As mentioned in section 1.1, the different integration technique used for the distance calculation also lead to slight discrepancies, but has considerably helped speed up the Markov chain Monte Carlo (MCMC) algorithms used to find the high scores. The MCMC currently iterates over multiple chains for around 10^6 points to output the high score, and each run takes around 30 minutes. Therefore, the original integration method would have been unreasonably slow. Other than these two changes, no differences between the two versions of the same game have been found, suggesting that Space Py Quest should work as expected.

3.3   Optimal Parameters

Another way of testing Space Py Quest is to check that the parameters achieving the maximum number of detections are realistic using an optimisation method. The expected optimal parameters for maximising the total number of source detections is compared to values output by the optimisation method, and are found to be as expected. This means that parameters corresponding to high scores for this game are also as expected.

Due to the fact that multiple parameters can be varied in Space Py Quest, a Markov Chain Monte Carlo method (specifically the Metropolis Hastings algorithm [10, 11]), was chosen. This algorithm is known for being efficient in multidimensional space, especially in comparison to grid search methods. It is typically used to sample from probability distributions; however, a variation of this method was implemented to obtain the high score. This in turn would output the best detector design while allowing us to test the performance of Space Py Quest. The *scipy.optimise.differential_evolution* [12] function was also used to ensure that the algorithm returned accurate results.

Markov Chain Monte Carlo

All parameters except detector location and mirror material were varied in the Markov Chain Monte Carlo (MCMC). Location and mirror material are categorical variables and have no well-defined incrementation, meaning that they cannot be varied in the MCMC. This is because a new value cannot be proposed depending on the current value of these variables, meaning it cannot form a Markov chain. Therefore, these parameters were set

arbitrarily to Jungle and Silicon respectively, and the frequency range set was to the aLIGO frequency band of $(1\text{-}10^4)$ Hz. An outline of the implementation method can be seen in algorithm 1.

Algorithm 1 MCMC algorithm

Start at random number of detections $N_{\text{Total}}(d, N_p, T, N_s, l, M, P, R)$
for i = 1 to number of iterations **do**
$\quad$ Propose new number of detections $N_{\text{Total}}(d', N_p', T', N_s', l', M', P', R')$
$\quad \alpha = N_{\text{Total}}(d', N_p', T', N_s', l', M', P', R')/N_{\text{Total}}(d, N_p, T, N_s, l, M, P, R)$
$\quad$ **if** $\alpha \geq u[0,1]$ **then**
$\qquad N_{\text{Total}}(d, N_p, T, N_s, l, M, P, R) = N_{\text{Total}}(d', N_p', T', N_s', l', M', P', R')$
$\quad$ **end if**
$\quad$ Record $N_{\text{Total}}(d, N_p, T, N_s, l, M, P, R)$
end for

The algorithm initialises with random variable values corresponding to a random number of total detections N_{Total}, calculated using the equations in section 2. It then proposes a jump to a nearby point for all parameters (the proposed values are indicated by the primes in algorithm 1). The proposal jump was chosen to be drawn from a gaussian distribution centred on the previous point, meaning that although the jump width varies, the proposed jump is most likely to be close to the current point. Subsequently, an acceptance ratio α is calculated, using the proposed and current position. If this ratio is greater than or equal to a uniformly drawn random number between 0 and 1, then the proposed step is accepted and the proposed number of detections becomes the current number of detections. Otherwise, the proposed step is rejected, and the chain remains where it is.

The use of the acceptance ratio means that the chain will always accept a move towards a higher number of detections, but also occasionally towards a lower number, depending on the ratio of the number of detections at the respective positions. This allows the search to move away from local maxima and ensures it finds the global maximum. At the end of each iteration, the current number of detections and corresponding parameters are recorded.

Finding the Maximum Number of Detections

As a simple test, the budget and complexity in the game were ignored, which meant that the effect of the parameters on the noise curves could be considered without being limited by their cost or complexity. According to the noise equations used, minimising noise should correspond to maximising mirror mass, number of stages and pumps, depth of the detector and suspension length, as well as minimising temperature and roughness of the mirrors. Increasing power increases radiation pressure noise but decreases shot noise, meaning a compromised value between the two limits is expected to be optimal.

Both optimisation methods output a maximum of 393 detections. The MCMC was ran for approximately 1×10^6 points, outputting 85 different combinations of parameters. This maximum could only be found when mirror mass was at its higher limit of 100kg, and roughness at its lower limit of 1nm. The optimal number of pumps ranged from 9km^{-1} upwards, and the number of stages from 4 upwards. The suspension length too varied in the higher end of its limit, from approximately 4km upwards, and the depth ranged from around 600m to 700m. This was expected, as a high value for all these parameters decreases noise in the game. The range of these optimal values can be explained by the fact that at peak sensitivity for total detections, the detector in the game is limited by radiation pressure and shot noise (see figure 8). These noises have the same quantum origin, and cannot be decreased simultaneously due to Heisenberg's uncertainty principle. This limit is called the *Standard Quantum Limit* [13]. The algorithm finds the optimal value for power varies between 35W and 45W, and cannot reduce both noise curves any further. It was discovered that at these parameters, the optimal value could be found at a large range of depths, from about 50m onwards. As the seismic and gravity gradient noises are well below the limiting curves, digging beyond 50m does not improve the overall detector sensitivity. The same reasoning can be used to explain why not all parameters are at their extreme limits for minimising individual noise curves.

An optimisation including the budget and complexity was also implemented, returning 203 detections. As expected, this was found at a combination compromising truly optimal values, due to the added limits. The true
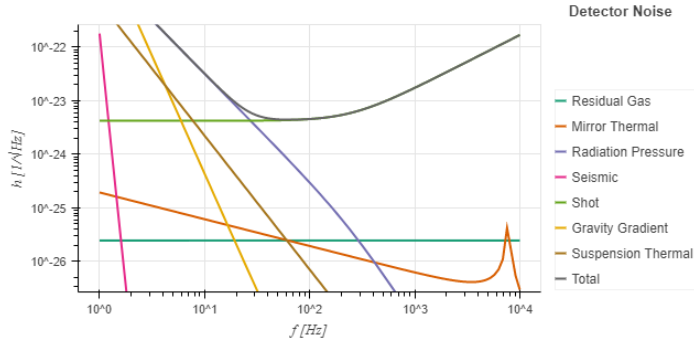
Figure 8: One combination of parameters that return maximum sensitivity without budget or complexity limits. The quantum noise curves limit detector sensitivity, meaning that parameters not included in these noises can vary over a wide range without changing the overall detector sensitivity.

parameters achieving this high score will not be revealed in this document.

3.4 Effects of Mirror Material and Location Choices

As location and mirror material were not varied in the MCMC, the effects of these variables were tested independently.

Mirror material: Silicon, Silica and Sapphire have the same values of mechanical damping factor, $\mathcal{L}$, but are distinguished by differing optical losses. This is outlined in appendix A.1. Lower optical loss means greater mirror quality factor $\mathcal{Q}$, leading to lower mirror thermal noise. This increases the sensitivity of the detector, leading to a greater detector range and number of detections.

According to this logic, Silicon should return the greatest detector range, followed by Sapphire and then Silica. Crystal appears to have the greatest optical loss, which leads to significantly greater mirror thermal noise as shown in figure 10. It also has a lower value of $\mathcal{L}$, which increases shot noise and reduces radiation pressure noise slightly, as well as reducing complexity. Consequently, selecting this material should lead to the lowest detector range and therefore number of detections. This was tested by setting the parameters to those for the aLIGO model (including Island for location), and varying the mirror materials. The results are detailed in table 2, confirming what is expected.

Table 2: Results obtained for different material choices with the aLIGO model. Silicon produces the greatest detector range, followed by Sapphire and Silica. Crystal clearly leads to the worst overall detector sensitivity, outputting no detections.

Material:	Sapphire	Crystal	Silicon	Silica
N_{bhbh}	2	0	2	2
N_{nsns}	6	0	6	5
N_{sn}	1	0	1	1
r_{bhbh} **(Mpc)**	653.07	58.57	657.35	644.92
r_{nsns} **(Mpc)**	138.54	4.95	142.74	131.37
Score (Mpc)	153.16	7.67	157.15	146.35
Total Cost ($)	68879252	57939252	63999252	61979252
Total Complexity	18.02	17.42	18.02	18.02

Location: The values correspond to location are detailed in appendix A.1. Desert has the least mechanical susceptibility scaling X_{dc} and high frequency floor X_{hf}, as well as the greatest exponent to the frequency-dependent scaling n_0. It is based on a mixed spectrum of globally seismic sites. According to equation 1, this location should result in the lowest values of seismic and gravity gradient noise. By contrast, City has high values of X_{dc} and $X_h f$, as well as the lowest value of n_0, meaning that it should output the maximum seismic and
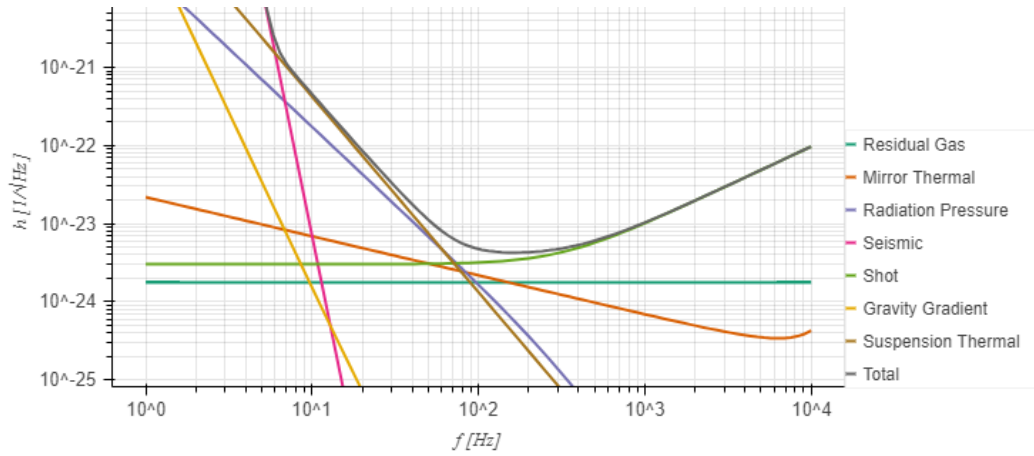
(a) Silicon
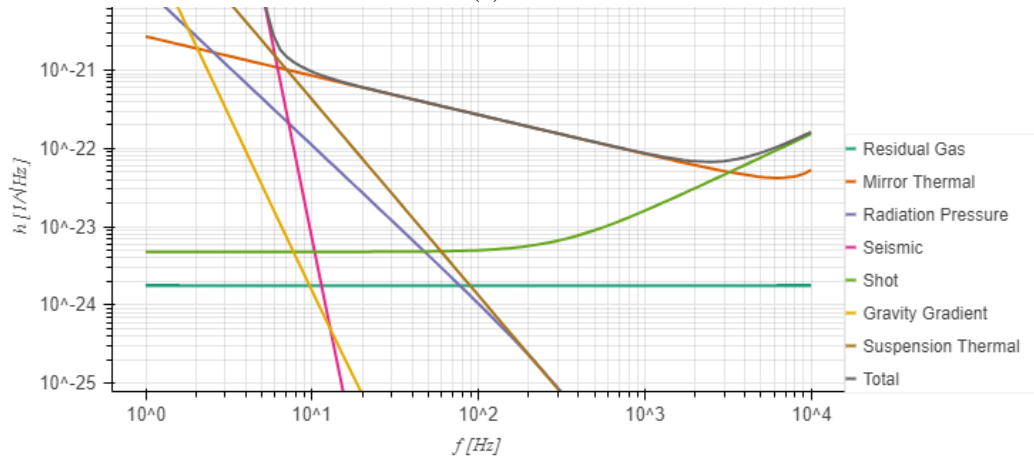
(b) Crystal

Figure 9: Comparison between effects of Silicon and Crystal mirror materials on the detector sensitivity. The selection of Crystal greatly increases mirror thermal noise, in turn decreasing the sensitivity of the detector across a large range of frequencies. It also slightly reduces radiation pressure noise and increases shot noise.

gravity gradient noise and therefore smallest detector range. Island and Jungle fall in the middle of these two location in terms of minimising noise. Island has a lower value of X_{dc} and higher n_0 than Jungle, but also a greater X_{hf}. This makes it difficult to deduce which of the two would result in a greater detector range.

Again setting the initial values to the aLIGO model (including Silica for mirror mass), the location was varied, and the results can be seen in table 3. The results for Island are the same as for Silica above as the parameter combinations are identical, but it is repeated for completeness. As expected, Desert and City return the greatest and smallest detector range respectively. The detector ranges for Jungle and Island fall in the middle as predicted, with Jungle appearing to be a slightly better option as it gives a greater detector range. The total number of detections does not vary as the ground motion noises are predominant at low frequencies and does not have a large effect with the selected source masses. A comparison between the Desert and City locations can be seen in figure 10b. Setting the City location increases seismic and gravity gradient noises, leading to a lower detector sensitivity.

Table 3: Results obtained for different location choices. As change in location only has an effect at low frequencies, the total number of detections remains the same at all locations. However, the Desert location maximises the range of the detector, followed by Jungle then Island. The high ground motion noise with City means it has the smallest detector range.

| Site:	City	Jungle	Desert	Island
N_{bhbh}	2	2	2	2
N_{nsns}	5	5	5	5
N_{sn}	1	1	1	1
r_{bhbh} **(Mpc)**	643.51	645.03	645.16	644.92
r_{nsns} **(Mpc)**	131.34	131.37	131.38	131.37
Score (Mpc)	146.26	146.36	146.36	146.35
Remaining Budget ($)	33020748	63020748	23020748	43020748

The complexity of the materials and the costs for both match up to the values selected for the game outlined in the appendices (the complexity does not change with location, it remains constant at 18.02).

3.5 Frequency Band Sensitivity

Gravitational wave detectors can be optimised for different frequency bands to detect selected sources. Currently, detector sensitivity is greatest in the range of $(10\text{-}10^3)$Hz. Compact binary inspirals release gravitational waves in the form of sinusoids, increasing in frequency and amplitude until the end of the inspiral phase. They spend a variable amount of time in different frequency bands. A neutron star binary inspiral, with each neutron star assumed to be a mass of $1.4\text{M}\odot$, will have a maximum gravitational wave frequency of 1500Hz [14]. Inspirals of higher mass terminate at proportionally lower frequencies. As neutron stars are less massive than black holes, setting parameters to achieve greater sensitivity at a high frequency band should return more detections of neutron stars. This property was checked for Space Py Quest as described below.

Again setting the parameters to mirror the aLIGO model, the power was increased to 200W. This decreased shot noise, which dominates at higher frequencies, and returned detections of five neutron star binaries, one black hole binary and one supernova.

The parameters were then varied to favour greater sensitivity at a lower frequency. This involved moving the power down to 7W to increase shot noise and reduce radiation pressure, changing location to Desert to reduce seismic and gravity gradient noise, and increasing suspension length and mirror mass to their maximum values, mostly to reduce radiation pressure and suspension thermal noises. The resulting noise curves can be seen in figure 11. This output three neutron star binaries, 32 black hole binary and zero supernova detections. As one supernova detection has been included for a good mid-high frequency range sensitivity, Space Py performs as expected in different frequency bands.
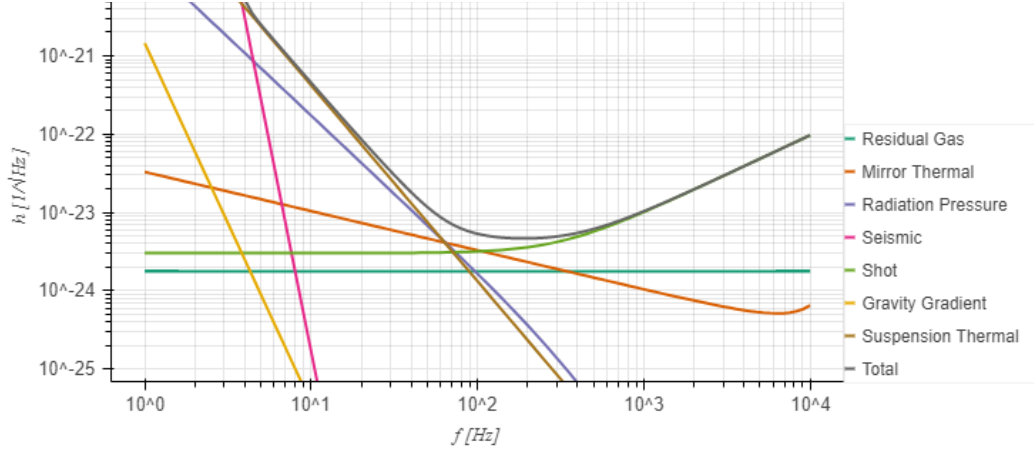
(a) Desert
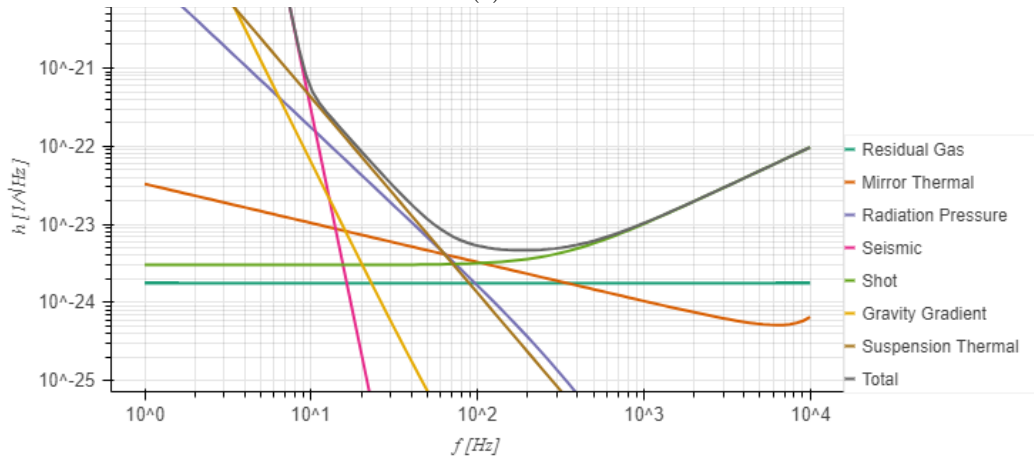
(b) City

Figure 10: Comparison between effects of Desert and City locations on the detector sensitivity. Selecting City leads to an increase in both seismic and gravity gradient noise. However, the effect on the total noise curve is over a small range at a low frequency, meaning that the overall detector range and sensitivity does not alter significantly

3.6   Conclusions of Testing

The primary checks carried out on Space Py Quest all appear to indicate that it behaves as intended. The discrepancies found between the original Space Py Quest have been investigated and documented, and the output noise curves scale with the relevant parameters as expected. The parameters outputting the maximum sensitivity for total number of detections is also as expected from the noise equations used in the game. Finally, it detects more neutron star binaries for a greater sensitivity at high frequency, and more black hole binaries for a greater sensitivity at a lower frequency, as it should.
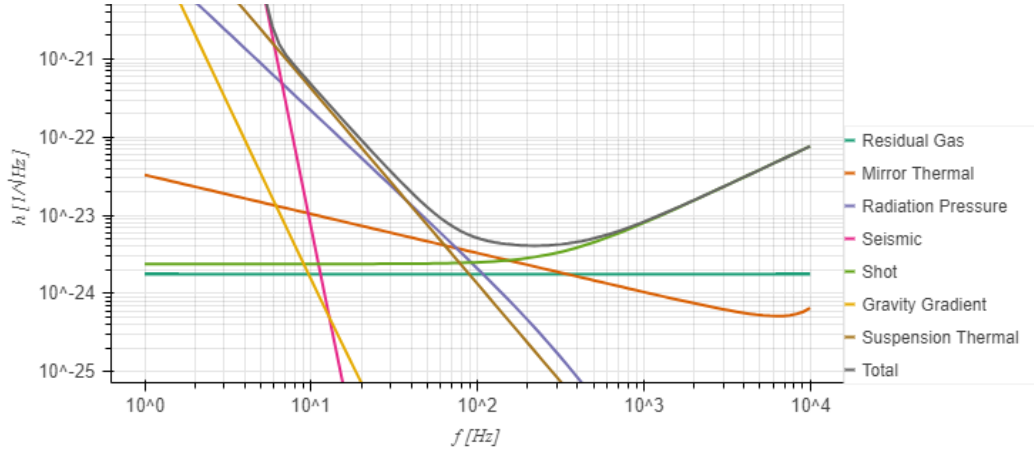
(a) Optimising for high frequency range
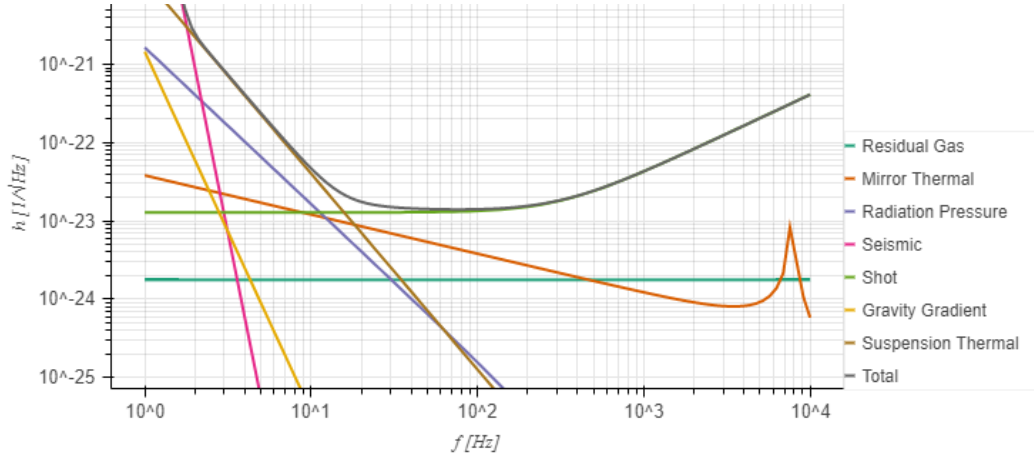
(b) Optimising for low frequency range

Figure 11: Sensitivity curves obtained when optimising for different frequency ranges. Optimising for higher frequencies means that high frequency sources like neutron stars and supernovae are more likely to be detected. Black holes signals terminate at lower frequencies, making them more likely to be found when optimising for low frequency.

4 Making Space Py Quest
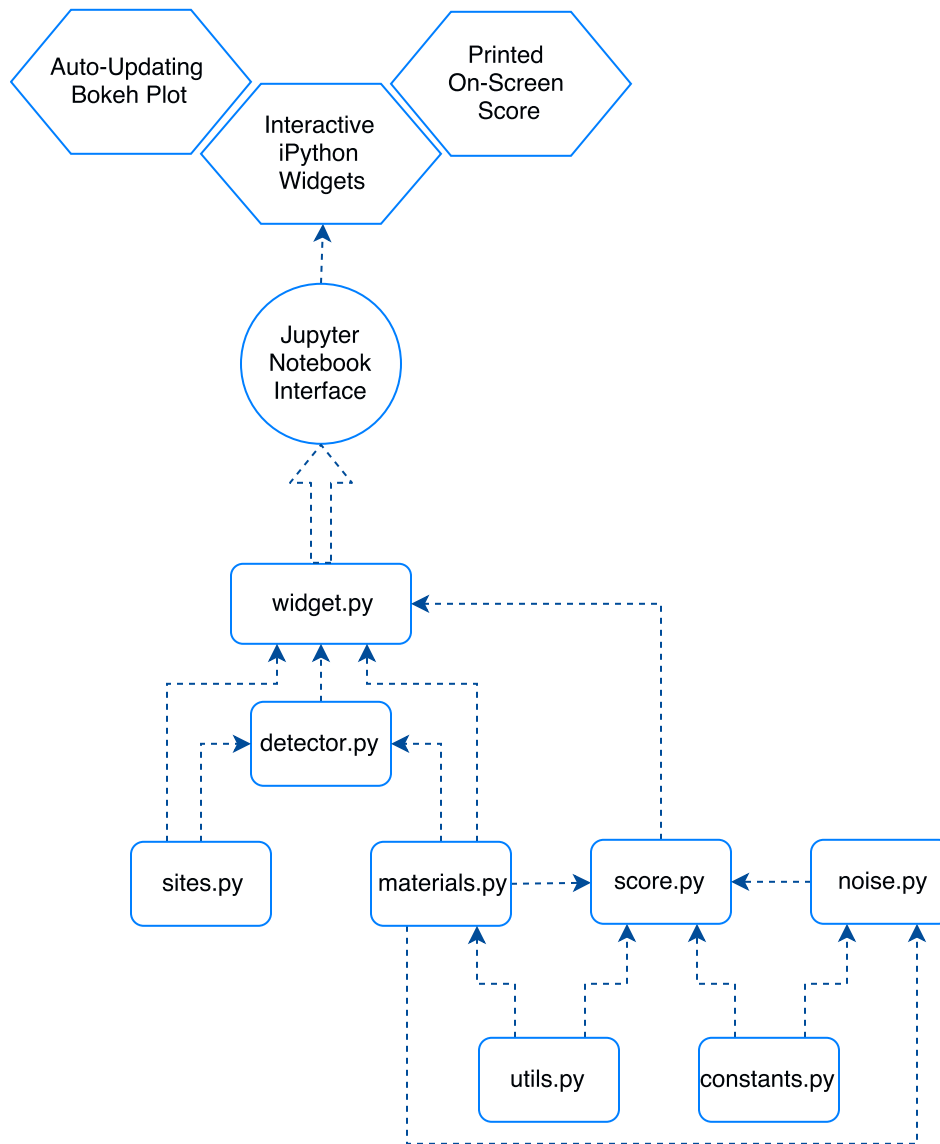
4.1 Code Structure

Figure 12: Code structure diagram for Space Py Quest. Utility functions and constants, defined in the appendices of this document, are held in *utils.py* and *constants.py* respectively. These are used throughout the rest of the game code. All other *.py* files displayed define the game classes. The **Detector** class is defined in *detector.py*. This imports *sites.py* and *materials.py* so that it can initialise its own location and mirror material parameters if the user does not provide them. The game interface is managed by the **spaceTimeQuest** class, which both handles the interaction of the iPython widgets with the Bokeh.io plot, and the interaction of the **Score** class with the detector object. **Score** and **ScoreCalculator** are defined in *score.py*, which calculate the figures of merit for the detector they are passed. They utilise all of the noise classes defined in *noise.py* to generate and return the detector's noise curves, cost, complexity, range, and the number of detections of different types of event.
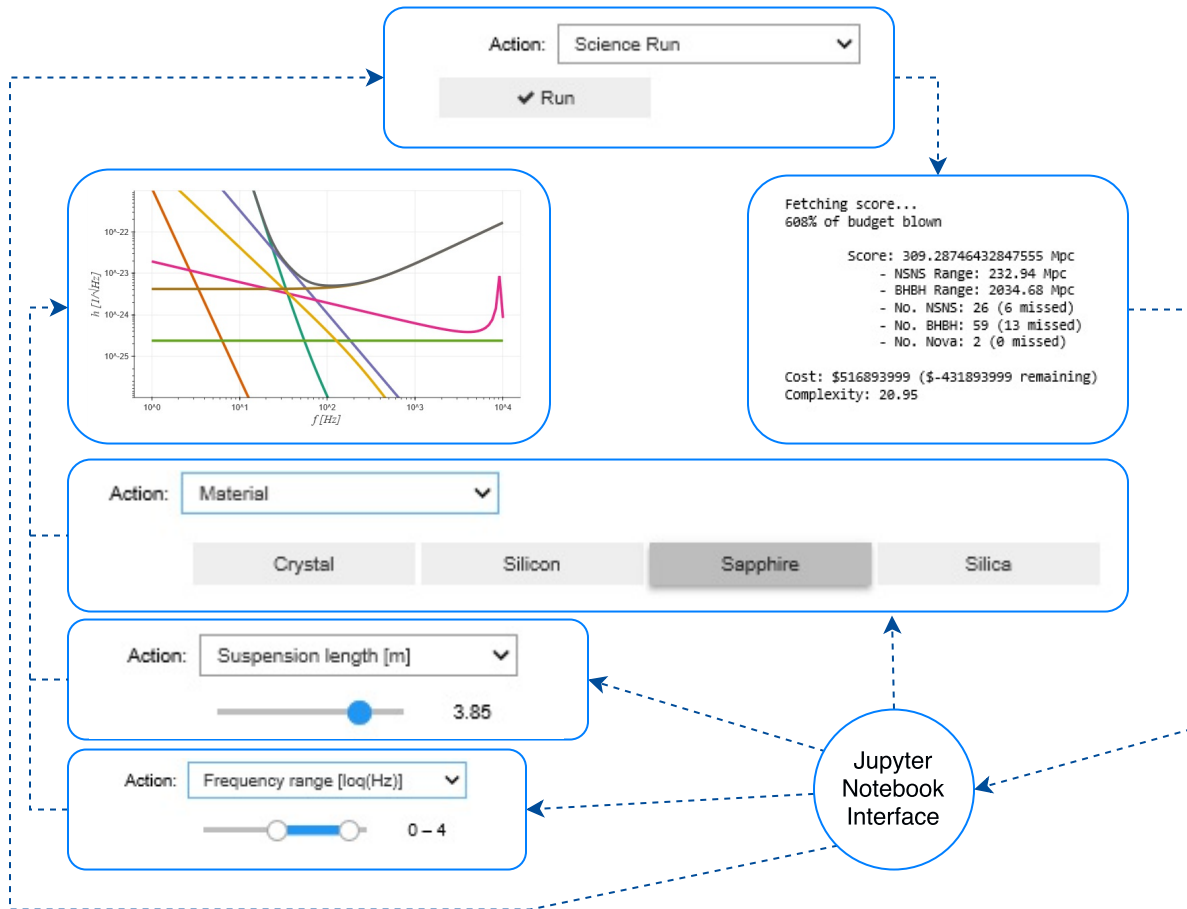
Figure 13: Illustration of some available widget types and the interactive output, both existing within the Jupyter Notebook interface. The categorical variables of location and mirror material are changed using toggle buttons. Continuous variables, like suspension length and mirror surface roughness, are altered using float-type slides. Discrete variables like the number of suspension stages are incremented using similar, integer-type sliders. The ranges displayed on both axes are varied using float-type range sliders. There are also tick boxes - not illustrated - which allow the user to determine which noise curves are displayed. The score calculation and display is triggered by a press of the *Run* button, which represents the detector activating a 'Science run'.

Space Py Quest is written in Python. The driving code has a semi-object-oriented structure, as illustrated and described in figure 12. The interactive elements of the game, existing in the Jupyter Notebook, are enabled using iPython Widgets, with a Bokeh.io plotting mechanism. iPython Widget examples are shown and described in figure 13. In addition to these, there are also Tick Boxes that determine which noise curves are shown. Each widget responds to user interaction, triggering recalculation of the noise curves and updating the displayed plot accordingly. The fraction spent of the user's budget is also continuously updated as the detector parameters change.

Suggestions for Structure Modifications

As shown in figure 12, the *materials* and *sites* classes must be imported by both *detector* and *widget*. This is because *widget* must know about these classes in order to set detector values, and *detector* must know about them in order to initialise itself with values, if none are provided by the user. Space Py Quest was designed to be adaptable to detectors that might have different sites or materials, but this is not necessarily a useful feature. It is proposed that instead, all parameter ranges and class names are stored in *constants*, and passed through *detector* to *widget*. Alternatively they could be passed straight to *widget*, without the option for the detector to initialise its own values.

5 Discussions and Outlook

5.1 Ideas for the Future

Space Py Quest was built in a limited time frame and is a work in progress. Some thoughts on possible steps forward are detailed below.

1. **Narrative**
   It is realistic that detector designs be swiftly modified and the results of these modifications considered. It is somewhat realistic that a detector could be modified after one observing run in order to make improvements. However, it is not realistic that the whole kilometer-scale detector could be picked up and moved to a different location. In order to make Space Py Quest's narrative and aims more similar to those of Space Time Quest, future versions of the game could limit users to 5 or so 'upgrades'. Additionally, putting the location specification widget in a preceding Jupyter notebook container would add some semblance of a one-way narrative.

2. **Themed Parameters**
   During the 'design phase' of Space Time Quest, the user can switch between 3 sets of parameters, each with a certain theme or 'subsystem': Environment, Vibration Isolation and Optics. Space Py Quest currently has just one drop-down menu from which all of the parameters can be accessed. This was done to test user addition of additional variables. It was also extremely easy to do using the dictionary of detector parameters. The parameters could instead be held in themed tabs to segregate them into subsystems, making the interface more similar to Space Time Quest.

3. **Leaderboard or Prize**
   There is not the same motivation for a user to obtain the largest range in Space Py Quest as there is in Space Time Quest. There is only scientific curiosity, which is perhaps *more* well-satisfied in Space Py Quest, as the result of the science run is more informative about specific sources. To be used a teaching tool for younger students, there could perhaps be a more accessible motivation, like a leaderboard or a printable certificate.

4. **Showcasing Resonance**
   Including slightly more accurate noise equations for things like mirror thermal noise would enable the curves to spike at resonant frequencies. The origins of these spikes may be too advanced for certain levels of education, but could be something to investigate when used as a teaching tool, as it has a visual effect on the noise curves that may be peculiar to someone unfamiliar with the underlying physics.

5. **User Definition of New Noise**
   In the package containing Space Py Quest is a file named *translate.py*. The functions in this file create a
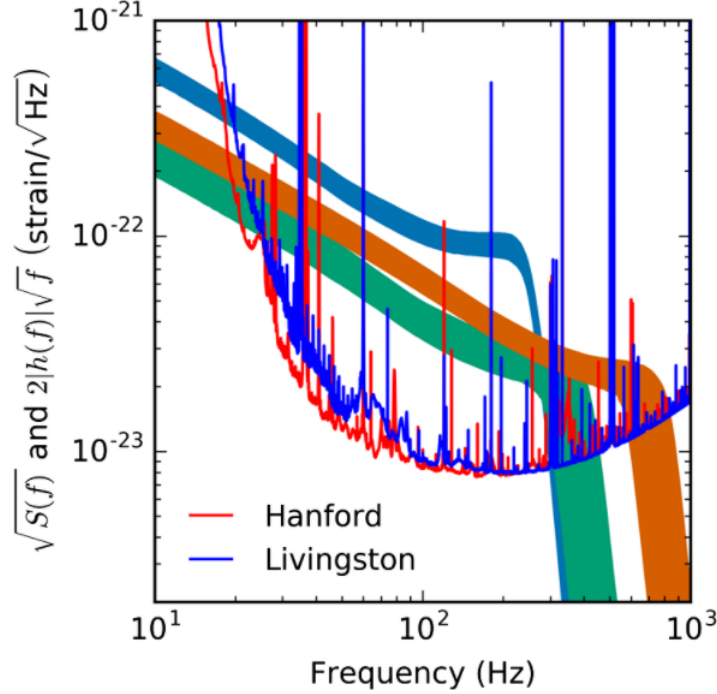
Figure 14: Frequency domain signals for detections GW150914 (blue), LVT151012 (green) and GW151226 (orange). These exact signals are nontrivial to recreate, but can be approximated fairly simply by splitting into 3 sections with different gradients. [15]

key to define new noise functions in terms of detector variables, generate new noise classes in a separate script, and import these new classes so that they can be passed to the score calculator. The new noise curves are then added to the existing noise plot, with an embedded, user-defined tag to denote their nature. This does not make up part of the game yet, but could be investigated by users, or included at a later date.

6. **Detector Distance Calculation**

The detector distance calculation considers the inspiral section of binary merger signals only. This means that the high-frequency sensitivity of the detector is actually irrelevant to the range reported, which is not physically realistic, and could end up teaching users incorrect physics. The shape of the frequency-domain signal could be approximated for a game like Space Py Quest very simply.

Frequency domain signals from the first 3 detections made by the LIGO collaboration are shown in figure 14. Phenomenological models, such as IMRPhenomB, generate very nearly accurate frequency domain gravitational waveforms. Doing this within Space Py Quest would be considerably more complex than required to provide a relatively realistic estimate of a detector range calculation.

The IMRPhenomB model proposed by Ajith *et al.* [8] uses three transitional frequencies to indicate at which stage in the merge a point within the binary signal has been emitted. Each make use of a conversion factor,

$$conv = \frac{GM_\odot}{c^2}. \tag{57}$$

The first, f_{merge}, describes the point at which the inspiral stage becomes the actual coalescence. The signal gradient up until this point is well-approximated within Space Py Quest. This can be simplified for the game from its value in Ajith:

$$f_{\mathrm{merge}} = (1 - 4.455 + 3.521) \times conv \approx 0.07 \times conv. \tag{58}$$

The coalescence moves from the merge to the ringdown stage at a frequency

$$f_{\mathrm{ring}} = \frac{1 - 0.63}{2} \times conv \approx 0.19 \times conv, \tag{59}$$

26

and the cutoff frequency can be given by

$$f_{\text{cutoff}} = 0.3236 \times conv \approx 0.32 \times conv. \tag{60}$$

Using these transition frequencies, a very simple inspiral waveform can be constructed using the shallow inspiral slope already calculated in Space Py Quest until f_{merge}, a horizontal line at this amplitude from f_{merge} to f_{ring}, and then a steeper slope from f_{ring} to f_{cutoff}.

5.2   A Personal Note

It may interest a reader to know that Space Py Quest was originally built as a training exercise towards the creation of a software package for performing full, non-simplified noise calculation for ground-based gravitational wave interferometers, called MAGIC. To this end, the making and testing of Space Py Quest has been an invaluable learning curve. We have discovered the benefits of using Python's *dictionary* type to make dynamic, quickly-modified models whose parameters are portable and easily inserted into testing methods. This has directly translated into the setup of our detector classes in MAGIC. Python's capacity for list comprehension has also significantly influenced our approach to our new noise calculation software, which we intend to be as efficiently written as possible. Space Py Quest is essentially a toy model of MAGIC, and the lessons learnt in constructing and using the game have both consciously and unconsciously weighted the procedures with which we construct and test the latter.

A   Appendix

Any symbols used within equations in this document that have a fixed value throughout the game are given in this section, in addition to any other functions that have not yet been noted.

A.1   Functions, Constants and Parameters

Utility Function

There is only one utility function, which returns the result of a linear interpolation between points $y(x)$ at point $x = t$. If t falls between points x_i and x_{i+1}, then the returned value is

$$\mathcal{I}(x, y, t) = y(t) = \frac{y_i(x_{i+1} - t) + y_{i+1}(t - x_i)}{x_{i+1} - x_i}. \tag{61}$$

Global Constants

Name	Symbol	Value
Speed of light	c	299792458 ms^{-1}
Newton's gravitational constant	G	6.67408×10^{-11}
Gravitational acceleration at Earth	g	9.81 ms^{-2}
Mass of the Sun	$M_\odot$	1.99×10^{30} kg
Planck's constant	h	6.626068×10^{-34} kgm^{2}s^{-1}
Boltzmann constant	k_b	1.380650×10^{-23} JK^{-1}
Atmospheric pressure	$\mathcal{P}$	1013 mbar
Temperature of Nitrogen	T_N	77 K
Radius of the Earth	R_E	6400 km
Astronomical unit	au	149598000 km
Megaparsec	Mpc	$3.08568025 \times 10^{22}$ km

Detector Constants

Detector parameters that cannot be altered through the Jupyter notebook interface are provided below, with examples are given from Virgo and/or aLIGO [5, 6].

Name	Symbol	Value	Note
Detector arm length	L	5000 m	Virgo arm length: 3000 m
			aLIGO arm length: 3994.5 m
Detector finesse	$\mathcal{F}$	60	Virgo finesse: 50
			aLIGO finesse : 450
Laser wavelength	λ	1064×10^{-9}	Virgo laser wavelength: 1064×10^{-9} m
			aLIGO laser wavelength: 1064×10^{-9} m
Detector power recycling factor	F_{PR}	100	Virgo power recycling factor: 50
			aLIGO power recycling factor: 43.61
First mirror resonance	f_R	4000 Hz	Virgo first mirror resonance: ≈ 5600 Hz
Cost of each vacuum pump	C_v	$850000	
Initial ambient temperature	T_0	300 K	Virgo ambient temperature: 300 K
			aLIGO ambient temperature: 290 K
Temperature increase per kilometer	ΔT	30 K	
Depth array	$\mathbf{d}$	$[0, 10, 100, 500]$ m	
Complexity array	$\mathbf{Z}_d$	$[0, 1, 4, 6]$	

Ranges

User-modifiable variables must be restricted to within physically realistic limits. Those enforced by Space Py Quest are given in the table below. Some information is provided to justify the choice of range [5, 6].

Name	Symbol	Range	Note
Frequency range	f	$[10^{-4}, 10^5]$ Hz	aLIGO and Virgo frequency ranges:
			$\sim 10^0 - 10^4$ Hz
Detector burial depth	d	$[0, 1000]$ m	Both Virgo and aLIGO are above ground,
			but future detectors like the Einstein
			Telescope (ET) could be buried
			100 - 200 m underground [16].
Number of vacuum pumps	N_p	$[0, 16]$	The number of vacuum pumps influences
			the pressure in the interferometer arms,
			as calculated using equation 18.
Detector temperature	T	$[1, 330]$ K	Virgo suspension temperature: 300 K
			aLIGO suspension temperature: 300 K
Number of suspension stages	N_s	$[1, 9]$	Both Virgo and aLIGO have 4 suspension stages.
Suspension length	l	$[0.35, 5]$ m	Virgo suspension length: 0.7 m
			aLIGO suspension length: ~ 0.5 m
Mirror mass	M	$[5, 100]$ kg	Virgo mirror mass: ~ 20.4 kg
			aLIGO mirror mass: ~ 40 kg
Laser power	P	$[1, 200]$ W	aLIGO laser power: 125 W
Mirror surface roughness	R	$[1, 500]$ nm	aLIGO mirror surface quality: ~ 0.1 nm

Site-dependent Values

The detector site influences the seismic noise curve, budget and complexity. The seismic noise is an approximation of the noise curves shown in figure 7 of Ohashi *et al.* [17], which is reproduced in figure 15 in this document. *City*, *Island* and *Jungle* come close to the curves for Tokyo, Kamioka and the Black Forest in order, whilst *Desert* is a mixed spectrum of global seismically quiet sites. Each *site* class defines 6 identically-named data members. The
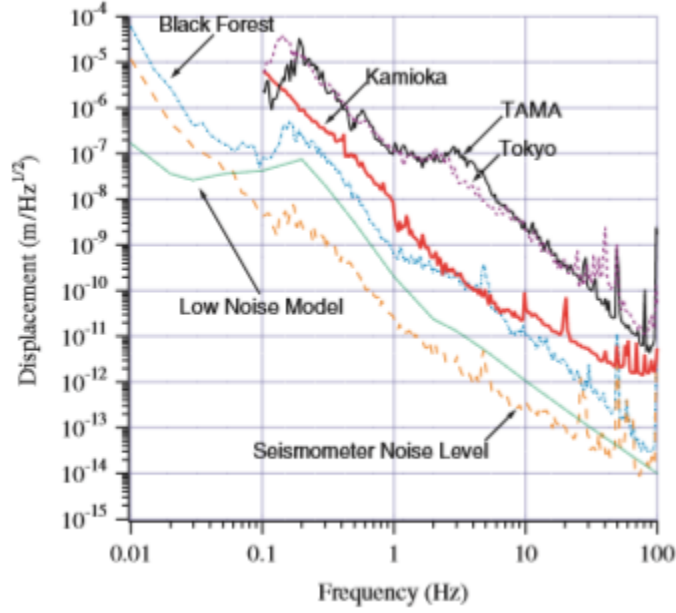
Figure 15: Seismic noise curves for Kamioka, TAMA300 site, Tokyo, Black Forest Geophysical Observatory in Germany, and a hybrid low noise model build from data from global quiet sites, as plotted by Ohashi *et al.* [17].

data members include *Complex Credits*, a scaling parameter for complexity, and *Budget*, the amount of money available to the player.

Name	Symbol	City	Jungle	Desert	Island
Complex Credits	Z_{cred}	20	19	17	18
Budget	-	95×10^6	125×10^6	85×10^6	105×10^6
Mechanical susceptibility scaling	X_{dc}	3×10^{-5}	5×10^{-5}	1×10^{-7}	8×10^{-6}
High-frequency floor	X_{hf}	1×10^{-11}	5×10^{-14}	8×10^{-15}	9×10^{-13}
Critical frequency	f_c	0.15	0.02	0.125	0.08
Exponent of frequency-dependent noise scaling	n_0	2.3	2.4	2.6	2.5

Material-dependent Values

Each material has a damping rate, $\mathcal{L}$. Materials with higher $\mathcal{L}$ contribute less to the mirror thermal noise. It should be relatively simple for a user to infer that Crystal is *not* a good mirror material to choose.

Name	Symbol	Sapphire	Crystal	Silicon	Silica
Goodness of losses	$\mathcal{L}$	1	0.4	1	1
Material base cost (\$)	C_0	4×10^6	5×10^5	2×10^6	1.5×10^6
Cost mass scaling (\$)	A_0	56×10^6	9.5×10^6	38×10^6	28.5×10^6
Temperature data (K)	$\mathbf{T}$	[1, 25, 80, 105, 230, 300]	[1, 300]	[1, 32, 40, 270, 300]	[1, 35, 90, 150, 200, 250, 300]
Losses data	$\mathbf{q}(T)$	[1.4e-9, 2.5e-8, 7e-9, 1.2e-8, 1.6e-8, 1e-7]	[1e-3, 1e-3]	[1.4e-9, 1.5e-8, 7.5e-9, 4.5e-8, 7e-8]	[1e-3, 7e-4, 1e-4, 3e-6, 3e-7, 1.5e-7, 1.5e-7]

The total mirror material cost is then

$$\mathcal{C}_{\mathrm{mat}} = C_0 + A_0 \left(\frac{M}{100} \right)^2. \tag{62}$$

A.2   Making Space Pie

Figure 16: The finished product, PieGO.

Space Pie Quest, or PieGO, is an interferometer made of pie. Specifically, apple strudel pie. It's made with ALDI ingredients, so it's cheap, and it's also vegan, and delicious. Thus, we maximise taste whilst minimising harmful consequences to the planet and our pockets.

Translation from non-vegan recipe

This pie is based on a recipe found here:
http://allrecipes.co.uk/recipe/16001/quick-apple-strudel.aspx. This uses an egg for glazing, which we replace with soy milk.

Pie Structure

First make your pie. Follow the instructions in the link given, and ensure that you make 2 long rectangular pies and 1 square one. At the assembly stage, wait until pies have cooled until configuring into a big 'L' shape with the square at the vertex. Congratulations; you have an interferometer-shaped pie. Dust this with icing sugar and serve.

Implementation

Eat, or enter (and win) a competition for 'best physics-themed bake'.

References

[1] D.D. Brown, P. Fulda, L. Carbone, P. Jones, and A. Freise. Space Time Quest, computer game for mobile devices and pc, 2009–. [website] Accessed 20.01.18 at https://www.laserlabs.org/spacetimequest.php. 1

[2] L. Carbone, C. Bond, D. Brown, et al. Computer-games for gravitational wave science outreach: Black hole pong and space time quest. *Journal of Physics: Conference Series*, 363:012057, 2012. 2

[3] Project jupyter. [website] Accessed 20.01.18 at http://jupyter.org/. 2

[4] ipywidgets. [website] Accessed 20.01.18 at https://ipywidgets.readthedocs.io/en/stable/. 2

[5] The LIGO Scientific Collaboration, J Aasi, B P Abbott, et al. Advanced ligo. *Classical and Quantum Gravity*, 32(7):074001, 2015. 5, 28

[6] M. Punturo. The virgo sensitivity curve. Technical Report VIR-NOT-PER-1390-51, 2004. 5, 6, 7, 9, 28

[7] D. H. Santamore and Yuri Levin. Eliminating thermal violin spikes from ligo noise. *Physical Review D*, 64(4), 8 2001. 7

[8] P. Ajith, M. Hannam, S. Husa, et al. Inspiral-merger-ringdown waveforms for black-hole binaries with nonprecessing spins. *Phys. Rev. Lett.*, 106:241101, Jun 2011. 9, 26

[9] LIGO Scientific Collaboration. GWINC: Gravitational wave interferometer noise calculator. http://ilog.ligo-wa.caltech.edu:7285/advligo/GWINC. 12

[10] N. Metropolis et al. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6), 1953. 16

[11] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1), 1970. 16

[12] E. Jones et al. SciPy: Open source scientific tools for Python, 2001–. [website] Accessed 20.01.18 at http://www.scipy.org/. 16

[13] Stefan L. Danilishin and Farid Ya. Khalili. Quantum measurement theory in gravitational-wave detectors. *Living Reviews in Relativity*, 15(1):5, Apr 2012. 17

[14] B P Abbott, R Abbott, R Adhikari, et al. Ligo: the laser interferometer gravitational-wave observatory. *Reports on Progress in Physics*, 72(7):076901, 2009. 20

[15] B. P. Abbott, R. Abbott, T. D. Abbott, et al. Binary black hole mergers in the first advanced ligo observing run. *Phys. Rev. X*, 6:041015, Oct 2016. 26

[16] ET science team. Einstein gravitational wave telescope conceptual design study. Technical Report ET-0106C-10, 2010. 28

[17] M Ohashi, K Kuroda, S Miyoki, et al. Design and construction status of clio. *Classical and Quantum Gravity*, 20(17):S599, 2003. 28, 29