PRECESSION. Dynamics of spinning black-hole binaries with python

Davide Gerosa[1, *] and Michael Kesden[2, †]

[1] *Department of Applied Mathematics and Theoretical Physics, Centre for Mathematical Sciences, University of Cambridge, Wilberforce Road, Cambridge CB3 0WA, UK*

[2] *Department of Physics, The University of Texas at Dallas, Richardson, TX 75080, USA*

(Dated: June 3, 2016)

We present the numerical code PRECESSION: a new open-source PYTHON module to study the dynamics of precessing black-hole binaries in the post-Newtonian regime. The code provides a comprehensive toolbox to (i) study the evolution of the black-hole spins along their precession cycles, (ii) perform gravitational-wave driven binary inspirals using both orbit-averaged and precession-averaged integrations, and (iii) predict the properties of the merger remnant through fitting formulae obtained from numerical-relativity simulations. PRECESSION is a ready-to-use tool to add the black-hole spin dynamics to larger-scale numerical studies such as gravitational-wave parameter estimation codes, population synthesis models to predict gravitational-wave event rates, galaxy merger trees and cosmological simulations of structure formation. PRECESSION provides fast and reliable integration methods to propagate statistical samples of black-hole binaries from/to large separations where they form to/from small separations where they become detectable, thus linking gravitational-wave observations of spinning black-hole binaries to their astrophysical formation history. The code is also a useful tool to compute initial parameters for numerical-relativity simulations targeting specific precessing systems. PRECESSION can be installed from the PYTHON Package Index and it is freely distributed under version control on GITHUB, where further documentation is provided.

PACS numbers: 04.25.dg, 04.25.Nx, 04.30.-w, 04.30.Tv, 04.70.Bw, 97.80.-d, 98.65.Fz

I. INTRODUCTION

Spinning black-hole (BH) binaries are remarkably interesting physical systems lying at the edge of fundamental physics and astronomy. Astrophysical BHs are described by the Kerr [1] solution of General Relativity and are fully characterized by their mass and angular momentum, or spin. In a binary system, couplings between the BH spins and the binary's orbital angular momentum introduce secular dynamical features on top of the binary's orbital motion: the two spins and the orbital plane precess about the direction of the total angular momentum of the system [2, 3]. Meanwhile, energy and momentum are slowly dissipated away in the form of gravitational waves (GWs) and the orbital separation consequently shrinks [4]. GW-driven inspiral may ultimately lead to the merger of the two BHs.

The three phenomena highlighted above (orbit, precession and inspiral) take place on different timescales. While the two BHs orbit about each other with period $t_{\rm orb} \sim (r/r_g)^{3/2}$, the spins and the orbital angular momentum precess at the rate $t_{\rm pre} \sim (r/r_g)^{5/2}$ and GW radiation reaction only affects the dynamics on times $t_{\rm RR} \sim (r/r_g)^4$ (here r is the binary separation and $r_g = GM/c^2$ is the gravitational radius of the total mass of the binary M). At separations $r \gg r_g$, the dynamics can be studied successfully using the post-Newtonian (PN) approximation to General Relativity (e.g. [5]) and the three timescales are widely separated: $t_{\rm orb} \ll t_{\rm pre} \ll t_{RR}$. Multi-timescale analyses can be used in this regime to efficiently disentangle the various dynamical features [3, 6, 7]. The timescale hierarchy breaks down, together with the entire PN approximation, at separations $r \sim r_g$ where the binary evolution can be followed faithfully only using numerical-relativity simulations (see e.g. [8]).

Spinning BHs now occupy a firm place in our understanding of the Universe. Astrophysical objects related to very energetic phenomena started being interpreted as BHs in the '60s [9, 10] following the identification of the first quasar [11] and the discovery of the first X-ray binary [12, 13]. BHs are observed in two separated mass regimes: stellar-mass BHs, which are the endpoints of the life of some massive stars [14], and supermassive BHs, which reside at the center of most galaxies and help regulate their evolution [15]. Although challenging, robust spin measurements from electromagnetic observations are now possible in both mass regimes [16, 17].

BHs have been long predicted to form binary systems: stellar-mass BH binaries are expected to form *in the field* from the evolution of massive binary stars [18] and dynamically in dense stellar clusters [19]; supermassive BH binaries are a natural by-product of hierarchical structure formation and galaxy mergers [20, 21]. BH binaries are now an observational reality. Following challenging electromagnetic observations (see e.g. [22] for a convincing candidate), the spectacular detection of GW150914 [23] from the LIGO interferometers [24] now constitutes irrefutable astrophysical evidence of a merging stellar-mass BH binary. Merging supermassive BH binaries are the main targets of the future space-based GW interferometer eLISA [25, 26] and current Pulsar Timing Arrays

* d.gerosa@damtp.cam.ac.uk

† kesden@utdallas.edu

[27–30].

Spin precession is a crucial ingredient to both BH physics and GW astronomy. Although precessional modulations in the emitted GW signal require development of more elaborate waveforms [31–33], they constitute a promising channel to extract astrophysical information from GW observations [34–36]. Moreover, PN spin precession introduces complex dynamics to the final stage of BH inspirals [37, 38] and greatly affects the properties of the BH remnants following binary mergers [39, 40].

In this paper, we present the numerical code PRECESSION: an open-source PYTHON module to study spinning BH binaries in the PN regime. In a nutshell, PRECESSION performs BH binary inspirals tracking their precessional dynamics using both standard orbit-averaged and new precession-averaged approaches. It also conveniently implements fitting formulae obtained from numerical-relativity simulations to predict mass, spin and recoil of BH remnants following binary mergers. PRECESSION combines the flexibility of the high-level programming language PYTHON with existing scientific libraries written in C and FORTRAN to bypass speed bottlenecks.

Our code finds application in a variety of astrophysical problems. Population synthesis models to predict GW rates (e.g. [41]) still lack the PN evolution of the BH spins which has been shown to critically depend on the binary formation channel [34]. Galaxy merger trees (e.g. [42, 43]) and large-scale cosmological simulations (e.g. [44, 45]) do not typically evolve the spin directions in the PN regime, although these are critical to address, e.g., the galaxy/BH occupation fraction [46, 47] and the detectability of recoiling BHs [48, 49]. We provide PN integrators to extend existing treatments of the astrophysical evolution of the BH spins [50–53] through the GW driven regime of the binary inspiral. The methods implemented in PRECESSION to analyze the BH spin dynamics could provide initial parameters to numerical-relativity simulations (e.g. [54, 55]) targeting specific precessing systems. GW parameter-estimation codes (e.g. [56]) may also benefit from our formulation of the spin-precession problem in terms of timescale separations. PRECESSION can easily propagate BH binaries backwards from GW observation to arbitrarily large separation, thus reconstructing their entire inspiral history. Our multi-timescale formulation of the problem could also help in the ongoing effort of building efficient GW templates for precessing systems [57]. Overall, we believe that PRECESSION will be a useful tool to interpret numerical results and GW observations of precessing BH binaries and facilitate more accurate modeling of their astrophysical environments.

This paper is organized as follows. Sec. II provides a general overview of the code; Sec. III is devoted to the spin precession dynamics; Sec. IV describes the integration of the PN equations of motion to perform BH inspirals; Sec. V summarizes the implementation of numerical-relativity fitting formulae to predict the properties of post-merger BHs; Sec. VI contains various practical examples to use PRECESSION; Sec. VII highlights our conclusions and anticipates future features of the code. From now on, equations are written in geometrical units ($c = G = 1$). As specified in Sec. II A, code units also set the binary's total mass to 1.

II. CODE OVERVIEW

In this section we give a general overview of the code. Sec. II A describes code installation; Sec. II B presents a minimal working example; Sec. II C provides details on documentation and source distribution; Sec. II D describes units and parallel programming features.

A. Installation

PRECESSION is a PYTHON [58] module and is part of the PYTHON Package Index: `pypi.python.org/pypi/precession`. The code can be installed in a single line through the package management system `pip`:

```
pip install precession
```

Useful options to the command above include `--user` for users without root privileges and `--upgrade` to update a pre-existing installation. The scientific libraries NUMPY [59], SCIPY [60], MATPLOTLIB [61] and PARMAP [62] are specified as prerequisites and, if not present, will be installed/updated together with PRECESSION. PRECESSION has been tested on PYTHON 2.7 distributions; porting to PYTHON 3 is under development.

Once PRECESSION has been installed, it has to be imported typing

```
import precession
```

from within a PYTHON console or script. The main module `precession` contains $\sim$ 80 functions for a total of $\sim$ 1700 code lines. The submodule `precession.test` consists of $\sim$ 300 code lines divided in 7 examples routines. If needed, this has to be imported separately typing

```
import precession.test
```

All functions and examples that should be called by the user are described in this paper.

B. A first working example

A minimal working example of some features of PRECESSION is shown in Fig. 1. We encourage the reader to execute this code snippet typing

```
precession.test.minimal()
```

We initialize a BH binary at the extremely large separation of 10 billion gravitational radii ($r = 10^{10}M$) and

Source code:

```
t0=time.time()
q=0.75    # Mass ratio
chi1=0.5  # Primary's spin magnitude
chi2=0.95 # Secondary's spin magnitude
print "Take a BH binary with q=%.2f, chi1=%.2f and
 ↪  chi2=%.2f" %(q,chi1,chi2)
sep=numpy.logspace(10,1,10) # Output separations
t1= numpy.pi/3.  # Spin orientations at r_vals[0]
t2= 2.*numpy.pi/3.
dp= numpy.pi/4.
M,m1,m2,S1,S2=precession.get_fixed(q,chi1,chi2)
t1v,t2v,dpv=precession.evolve_angles(t1,t2,dp,sep,q,S1,S2)
print "Perform BH binary inspiral"
print "log10(r/M) \t theta1 \t theta2 \t deltaphi"
for r,t1,t2,dp in zip(numpy.log10(sep),t1v,t2v,dpv):
    print "%.0f \t\t %.3f \t\t %.3f \t\t %.3f" %(r,t1,t2,dp)
t=time.time()-t0
print "Executed in %.3fs" %t
```

Screen output:

```
Take a BH binary with q=0.75, chi1=0.50 and chi2=0.95
Perform BH binary inspiral
log10(r/M)  theta1  theta2  deltaphi
10          1.047    2.094  -2.330
9           1.047    2.094  1.811
8           1.047    2.095  2.341
7           1.046    2.095  2.827
6           1.050    2.093  0.351
5           1.055    2.089  -0.211
4           1.046    2.095  -1.588
3           0.991    2.133  -2.271
2           0.909    2.190  -1.903
1           0.505    2.439  -1.188
Executed in 5.526s
```

FIG. 1. Source code (top) and screen output (bottom) of the example `test.minimal` described in Sec. II B. We select a BH binary at $r = 10^{10}M$ and track the directions of the two spins and the orbital angular momentum [cf. Eqs.(1)-(4)] during its PN inspiral till $r = 10M$. We use precession-averaged PN equations, which require random samplings of the precessional phase, see Sec. IV (different code executions will therefore return different values of the spin angles). The execution time reported is obtained using a single core of a 2013 Intel i5-3470 3.20GHz CPU. These lines can be executed typing `precession.test.minimal()`.

evolve it down to small separations ($r = 10M$) where the PN approximation breaks down. The integration is performed using precession-averaged PN equations of motion, as described later in Sec. IV B. The evolution of the BH spins along such an enormous separation range is computed in less than 6 seconds using a single core of a standard off-the-shelf desktop machine.

C. Documentation and source distribution

This paper describes the numerical code PRECESSION in its v1.0 release. The code is under active development and additional features will be added regularly. Earlier versions of the code were used in the following published results: [6, 7, 47, 63–66].

The source code is distributed under GIT version-control system at

`github.com/dgerosa/precession` (code),

and it is released under the CC BY 4.0 license. Extensive code documentation can be generated automatically in html format from the PYTHON's docstrings using the text processor PDOC [67]. Documentation is regularly uploaded to a dedicated branch of the GIT repository and it is available at

`dgerosa.github.io/precession` (documentation).

The same information can also be accessed using PYTHON's built-in help system, e.g. `help(precession.function)`. Additional resources and results are available at `davidegerosa.com/precession`.

D. Units and parallel features

All quantities in the code must be specified in total-mass units, i.e. $c = G = M = 1$. For instance, the code variable for the binary separation `r` stands for rc^2/GM; equivalently, the angular-momentum magnitude variable `L` stands for cL/GM^2.

PRECESSION includes some parallel programming features. *Embarrassingly* parallel tasks, such as computing several PN inspirals (Sec. IV), are sent to different cores to speed up the computation. By default, PRECESSION autodetects the number of available cores in the executing machine and splits the operations accordingly. Parallel execution can be controlled using the global integer variable `CPUs`, which specifies the number of parallel processes. For instance, serial execution can be enforced setting `CPUs=1` (cf. Sec. VI F).

Outputs of some functions are automatically stored, such that further executions of code scripts do not require full recalculation. The location of the output directory is controlled by the global string variable `storedir`, which is set by default to `"./precession_checkpoints"`. The output directory is automatically created if needed, or can be created manually using `make_temp`. Stored datafiles can be deleted using `empty_temp`.

III. SPIN PRECESSION

In this section we present how to use PRECESSION to study BH binaries on the spin precession timescale where GW emission can be neglected. After introducing double-spinning BH binaries (Sec. III A), we describe two useful parametrizations of the precession dynamics (Sec. III B) and discuss their constraints (Sec. III B). Time evolution of BH binaries along their precession cycles is described in Sec. III D. Finally, Sec. III E shows how to classify BH binaries according to their precessional morphologies.

A. Black-hole binaries in the post-Newtonian regime

Throughout this paper we only consider BH binaries on quasi-circular orbits. Astrophysical BH binaries are expected to circularize at large separation [4, 68] and the first GW detection confirms this finding [69]. However, eccentricity may be relevant for stellar-mass BH binaries formed in globular clusters [19] and supermassive BH binaries interacting with dense stellar environments [70, 71]. Generalization to eccentric orbits is an important extension of PRECESSION, which is left to future work.

We use standard notation where the component masses m_1 and m_2 are combined into total mass $M = m_1 + m_2$, mass ratio $q = m_2/m_1 \leq 1$ and symmetric mass ratio $\eta = m_1 m_2/M^2 = q/(1+q)^2$; the spin magnitudes $S_i = m_i^2 \chi_i$ (hereafter $i = 1, 2$) are given in terms of the dimensionless spin parameters $0 \leq \chi_i \leq 1$. The magnitude of the orbital angular momentum $\mathbf{L}$ is related to the binary separation r through the Newtonian expression $L = m_1 m_2 \sqrt{r/M}$. The utility `get_fixed` provides the component masses m_i and the spin magnitudes S_i in terms of q and χ_i in code units; similarly, `get_L` returns the Newtonian expression for the magnitude of the orbital angular momentum.

Before proceeding with the code implementation, we point out that PRECESSION is explicitly designed to handle genuine double-spin physics. Non-spinning and single-spin binaries (i.e. $\chi_1 = 0$ and/or $\chi_2 = 0$) represent singular cases that cannot be handled with the present version of the code. In practice, these systems can be well approximated by setting $\chi_i \gtrsim 0.001$.

PRECESSION loses accuracy in the extreme-mass-ratio limit $q \to 0$ (where other methods are required to study the dynamics, e.g. [72]) and the equal-mass limit $q \to 1$ [where the parametrization chosen to describe the precession cycle breaks down, e.g. Eq. (9)]. Our results have been well tested in the regime $0.005 \lesssim q \lesssim 0.995$. PRECESSION currently features an alternative implementation to study the strictly equal-mass case $q = 1$, which exploits additional constants of motion [73, 74]. These findings will be presented elsewhere [75].

B. Parametrization of double spin precession

The time evolution of the three vectors $\mathbf{S_1}$, $\mathbf{S_2}$ and $\mathbf{L}$ in an inertial frame is a nine-parameter problem. However, only four parameters are needed to describe the relative orientations of the three momenta [76–78]. One of these parameters is the orbital separation r (or equivalently the magnitude L), which is constant on $t_{\rm pre}$ and decreases on $t_{\rm RR}$ because of GW emission. Two possible choices for the remaining three degrees of freedom are:

1. The spin directions can be described in terms of three angles

$$\cos\theta_1 = \hat{\mathbf{S}}_\mathbf{1} \cdot \hat{\mathbf{L}}\,, \tag{1}$$

$$\cos\theta_2 = \hat{\mathbf{S}}_\mathbf{2} \cdot \hat{\mathbf{L}}\,, \tag{2}$$

$$\cos\Delta\Phi = \frac{\hat{\mathbf{S}}_\mathbf{1} \times \hat{\mathbf{L}}}{|\hat{\mathbf{S}}_\mathbf{1} \times \hat{\mathbf{L}}|} \cdot \frac{\hat{\mathbf{S}}_\mathbf{2} \times \hat{\mathbf{L}}}{|\hat{\mathbf{S}}_\mathbf{2} \times \hat{\mathbf{L}}|}, \tag{3}$$

where the sign of $\Delta\Phi$ is chosen such that

$$\operatorname{sgn}\Delta\Phi = \operatorname{sgn}\{\mathbf{L} \cdot [(\mathbf{S_1} \times \mathbf{L}) \times (\mathbf{S_2} \times \mathbf{L})]\}. \tag{4}$$

In words, θ_1 and θ_2 are the angles between the two spins and the orbital angular momentum (*tilt angles*) and $\Delta\Phi$ is the angle between the projections of the two spins onto the orbital plane (see Fig. 1 in [78]). Despite being very intuitive, this description makes the understanding of the underlying phenomenology rather complicated because all three variables $(\theta_1, \theta_2, \Delta\Phi)$ vary on both the precession and the inspiral timescales.

2. A more physical choice can be made to exploit the timescale separation $t_{\rm pre} \ll t_{RR}$. The magnitude of the total angular momentum

$$J = |\mathbf{L} + \mathbf{S_1} + \mathbf{S_2}| \tag{5}$$

is conserved on the timescale $t_{\rm pre}$ where GW emission can be neglected. Moreover, the projected effective spin [73, 79]

$$\xi \equiv M^{-2}[(1+q)\mathbf{S}_1 + (1+q^{-1})\mathbf{S}_2] \cdot \hat{\mathbf{L}}\,, \tag{6}$$

is a constant of motion of the (orbit-averaged) 2PN spin-precession and 2.5PN radiation-reaction equations (cf. Sec. IV A) and is therefore conserved on both $t_{\rm pre}$ and $t_{\rm RR}$. This implies that the entire dynamics on $t_{\rm pre}$ can be encoded in a single variable, which can be chosen[1] to be the magnitude of the total spin [6]

$$S = |\mathbf{S_1} + \mathbf{S_2}|\,. \tag{7}$$

The two descriptions –in terms of $(\theta_1, \theta_2, \Delta\Phi)$ and (ξ, J, S)– are related by the following sets of transfor-

[1] Equivalently, one can choose the angle φ' defined in Eq. (9) of [7]. PRECESSION contains additional routines to analyze the dynamics in terms of this angle. The most relevant functions are called `get_varphi` and `region_selection`.

mations

$$\begin{cases} S = [S_1^2 + S_2^2 + 2S_1S_2(\sin\theta_1\sin\theta_2\cos\Delta\Phi \\ \qquad\qquad + \cos\theta_1\cos\theta_2)]^{1/2} \ , \\ J = [L^2 + S^2 + 2L(S_1\cos\theta_1 + S_2\cos\theta_2)]^{1/2} \ , \\ \xi = \dfrac{1+q}{qM^2}(qS_1\cos\theta_1 + S_2\cos\theta_2) \ ; \end{cases} \tag{8}$$

$$\begin{cases} \cos\theta_1 = \dfrac{1}{2(1-q)S_1}\left[\dfrac{J^2-L^2-S^2}{L} - \dfrac{2qM^2\xi}{1+q}\right] , \\ \cos\theta_2 = \dfrac{q}{2(1-q)S_2}\left[-\dfrac{J^2-L^2-S^2}{L} + \dfrac{2M^2\xi}{1+q}\right] , \\ \cos\Delta\Phi = \dfrac{1}{\sin\theta_1\sin\theta_2}\left(\dfrac{S^2-S_1^2-S_2^2}{2S_1S_2} - \cos\theta_1\cos\theta_2\right) ; \end{cases} \tag{9}$$

which are implemented in `from_the_angles` and `parametric_angles`. Similarly, Eqs. (1)-(4) can be evaluated using `build_angles`. The angle $\theta_{12} = \arccos\hat{\mathbf{S}}_1\cdot\hat{\mathbf{S}}_2$ between the two spins can be computed using both sets of variables:

$$\begin{aligned} \cos\theta_{12} &= \frac{S^2-S_1^2-S_2^2}{2S_1S_2} \\ &= \sin\theta_1\sin\theta_2\cos\Delta\Phi + \cos\theta_1\cos\theta_2 \, . \end{aligned} \tag{10}$$

Eqs. (8) and (9) do not depend on the sign of $\Delta\Phi$. This reflects the symmetry of the dynamics between the first and second half of the precession cycle (cf. Sec. III D). If the spin vectors are available in the current computation (e.g. from orbit-averaged evolutions, see Sec. IV A), PRECESSION evaluates the sign of $\Delta\Phi$ directly from Eq. (4). If this is not the case, $\mathrm{sgn}\,\Delta\Phi$ must be specified by the user according to the evolution of S, as in the example of Sec. VI B. In case of precession-averaged inspirals (Sec. IV C), the sign of $\Delta\Phi$ is assigned randomly.

C. Geometrical constraints

The physical range of the three angles $(\theta_1, \theta_2, \Delta\Phi)$ is given by the independent constraints $\theta_1 \in [0,\pi]$, $\theta_1 \in [0,\pi]$ and $\Delta\Phi \in [-\pi,\pi]$. Geometrical constraints on ξ, J, S can be derived from Eqs. (6)-(7) and read:

$$-(1+q)(S_1 + S_2/q) \ \le \ M^2\xi \ \le \ (1+q)(S_1 + S_2/q) \, , \tag{11}$$

$$\max(0, L - S_1 - S_2, |S_1 - S_2| - L) \ \le \ J \ \le L + S_1 + S_2 \, , \tag{12}$$

$$|S_1 - S_2| \ \le \ S \ \le S_1 + S_2 \, . \tag{13}$$

Eqs. (11), (12) and (13) are returned by `xi_lim`, `J_lim` and `Sso_limits`, respectively. These constraints are not independent of each other. For a given J satisfying Eq. (12), the magnitude $S = |\mathbf{S_1} + \mathbf{S_2}| = |\mathbf{J} - \mathbf{L}|$ has to satisfy

$$\max(|J-L|, |S_1-S_2|) \le S \le \min(J+L, S_1+S_2) \, , \tag{14}$$

which is given by `St_limits`. Allowed values of ξ are then given by

$$\min_S \xi_-(S) \ \le \xi \ \le \ \max_S \xi_+(S) \, , \tag{15}$$

where $\xi_\pm$ are the *effective potentials* for BH binary spin precession [6]

$$\begin{aligned} \xi_\pm(S) = \{&(J^2-L^2-S^2)[S^2(1+q)^2 - (S_1^2-S_2^2)(1-q^2)] \\ &\pm (1-q^2)\sqrt{[J^2-(L-S)^2][(L+S)^2-J^2]} \\ &\times \sqrt{[S^2-(S_1-S_2)^2][(S_1+S_2)^2-S^2]}\}/(4qM^2S^2L) \, . \end{aligned} \tag{16}$$

In Ref. [7] we proved that ξ_+ (ξ_-) admits a single maximum (minimum) within the range of S given by Eq. (14) for any value of J satisfying Eq. (12)[2]. The extremization of the effective potentials is performed in `xi_allowed` using `scipy.optimize.fminbound` with a bracketing interval given by Eq. (14). Analogously, `J_allowed` computes the allowed range of J for any value of ξ satisfying Eq. (11). If needed, the effective potentials of Eq. (16) can be evaluated directly using `xi_plus` and `xi_minus`; their derivatives $d\xi_\pm/dS$ are implemented in `dxidS_plus` and `dxidS_minus`.

Once consistent values of J and ξ have been selected (cf. Sec. VI A for a practical example), the binary dynamics on $t_{\rm pre}$ is fully encoded in the evolution of S. The magnitude S oscillates between the two solutions $S_\pm$ of the equations $\xi_\pm(S) = \xi$. A precession cycle therefore consists of a complete oscillation $S_- \to S_+ \to S_-$. The radical equations $\xi_\pm(S) = \xi$ are solved in `Sb_limits` using `scipy.optimize.brentq`. From experiments in wide regions of the parameter space, we report a numerical accuracy of $\Delta S_\pm/M^2 \sim 10^{-8}$.

The two roots $S_\pm$ coincide at the extrema of the effective potentials $\xi = \min_S \xi_-(S)$ and $\xi = \max_S \xi_+(S)$, where consequently the magnitude of the total spin S remains constant. These are peculiar configurations where the *relative* orientation of $\mathbf{S_1}$, $\mathbf{S_2}$ and $\mathbf{L}$ does not evolve on $t_{\rm pre}$. It is straightforward to prove that they are characterized by $\sin\Delta\Phi = 0$: the three angular momenta share the same plane and jointly precess about the direction of $\mathbf{J}$. These solutions have been discovered more than a decade ago by Schnittman [76] and called *spin-orbit resonances* (for other studies see [80, 81]). One can prove that extremizing the effective potential $\xi_\pm$ is equivalent to solving Eq. (3.5) of [76]. Two spin-orbit resonances are present for any value of ξ: they are characterized by $\Delta\Phi = 0$ and $\Delta\Phi = \pi$ and correspond to the largest and lowest values of J compatible with the chosen ξ (cf. Fig. 5 in [7]). The angles θ_1 and θ_2 corresponding to both resonances $\Delta\Phi = 0, \pi$ can be evaluated using `resonant_finder`.

[2] One can also prove that $\min_S \xi_-(S) = \max_S \xi_+(S)$ if and only if $J = L + S_1 + S_2$ or $J = \max(0, L - S_1 - S_2, |S_1 - S_2| - L)$ [7]. Only one value of ξ is allowed in these peculiar cases and can be evaluated using `xi_at_Jlim`.

The values of J and ξ corresponding to the four (anti)aligned configurations $\cos\theta_i = \pm 1$ are returned by `aligned_configurations`. The thresholds of the precessional instability discovered in [63] are returned by `updown`.

D. Binary evolution on the precession timescale

The rate of variation of S between the two extrema $S_\pm$

$$\frac{dS}{dt} = -\frac{3(1-q^2)}{2q}\frac{S_1 S_2}{S}\frac{(\eta^2 M^3)^3}{L^5}\left(1-\frac{\eta M^2 \xi}{L}\right) \times \sin\theta_1 \sin\theta_2 \sin\Delta\Phi \tag{17}$$

$$= \pm\frac{3}{2}\eta M\left[1-\xi\left(\frac{r}{M}\right)^{-1/2}\right]\left(\frac{r}{M}\right)^{-5/2} \times \sqrt{(\xi_+ - \xi)(\xi - \xi_-)} \tag{18}$$

follows directly from the 2PN spin-precession equations [here reported in Eqs. (24-26), see [3]] and can be evaluated using `dSdt`. The solutions $S_\pm$ of the equations $\xi_\pm(S) = \xi$ correspond to turning points in the evolution of S, i.e. $dS/dt = 0$. The time evolution of a BH binary during (half of) a precession cycle is given by the integral

$$t(S) = \int_{S_-}^{S} \frac{dS'}{|dS'/dt|}\,, \quad S \in [S_-, S_+]\,. \tag{19}$$

The integrand $|dt/dS|^{-1}$ is regular everywhere in $S \in (S_-, S_+)$, while the limits

$$\lim_{S\to S_\pm} \frac{1}{|dS/dt|} \propto \frac{1}{\sqrt{|S - S_\pm|}} \tag{20}$$

ensure integrability[3] at $S_\pm$. The numerical integration of Eq. (19) is performed in `t_of_S` and its inverse `S_of_t`, using standard quadrature through `scipy.integrate.quad`. Eq. (19) can used to reparametrize the binary dynamics in terms of time (cf. Sec. VI B). The precessional period τ is defined as the time for a complete precession cycle $S_- \to S_+ \to S_-$

$$\tau = 2\int_{S_-}^{S^+} \frac{dS'}{|dS'/dt|}\,, \tag{21}$$

and can be computed using `precessional_period`.

The direction of $\mathbf{J}$ is constant as long as radiation reaction is being neglected. The orbital angular momentum $\mathbf{L}$ precesses about that fixed direction at a rate [6]

$$\Omega_z = \frac{J}{2}\left(\frac{\eta^2 M^3}{L^2}\right)^3 \left\{1 + \frac{3}{2\eta}\left(1-\frac{\eta M^2\xi}{L}\right) - \frac{3(1+q)}{2q}\left(1-\frac{\eta M^2\xi}{L}\right)[4(1-q)L^2(S_1^2 - S_2^2) - (1+q)(J^2 - L^2 - S^2)(J^2 - L^2 - S^2 - 4\eta M^2 L\xi)] \times [J^2 - (L-S)^2]^{-1}[(L+S)^2 - J^2]^{-1}\right\}. \tag{22}$$

The vector $\mathbf{L}$ therefore spans an angle

$$\alpha = 2\int_{S_-}^{S_+} \Omega_z \frac{dS}{|dS/dt|} \tag{23}$$

about $\mathbf{J}$ during each precession cycle. Eqs. (22) and (23) can be evaluated using `Omegaz` and `alphaz`, respectively. The azimuthal angle of the projection of $\mathbf{L}$ onto a plane orthogonal to $\mathbf{J}$ can be tracked using `alpha_of_S`, cf. Eq. (30) of [7]. The conditions $\alpha = 2\pi n$ (n integer) correspond to configurations where the precession frequency of $\mathbf{L}$ about $\mathbf{J}$ and that of the two spins are in resonance with each other [82]. Tools to analyze such peculiar configurations will be made available in future versions of the code.

E. Spin morphologies

As discussed at great length in [7], the precessional behavior of spinning BH binaries can be classified in terms of three different *morphologies*. These are related to the evolution of $\Delta\Phi$ during a precession cycle. In particular, three situations are possibile:

1. $\Delta\Phi$ circulates through the full range $[-\pi, +\pi]$;
2. $\Delta\Phi$ librates about 0 (and never reaches $\pm\pi$);
3. $\Delta\Phi$ librates about $\pm\pi$ (and never reaches 0).

Examples of BH binaries in the different morphologies are studied in Sec. VI A. The spin-orbit resonances $\xi = \min_S(\xi_-)$ and $\xi = \max_S(\xi_+)$ can be interpreted as the limits of the two librating morphologies: as the precession amplitude $(S_+ - S_-)$ goes to zero, $\Delta\Phi$ approaches one of the resonant configurations and locks onto either 0 or $\pm\pi$ [76]. The spin morphology is an interesting dynamical feature of BH binaries because, while it characterizes spin precession, it does not vary on the precession timescale (i.e., it is independent of S). Radiation reaction causes morphological transitions which are promising GW observables [65, 78]. Morphological classification is implemented in `find_morphology`.

The loop formed by the two effective potentials $\xi_\pm$ of Eq. (16) encloses all binary configurations (ξ, S) compatible with fixed values of r, J, q and S_i. Regions of binaries with different morphologies can coexist in this plane in the following way (see Fig. 4 of [7]):

[3] The only exception is the up-down configuration ($\cos\theta_1 = 1$, $\cos\theta_2 = -1$) in its instability region, where $\tau \to \infty$ [63].

1. a single region where all binaries librate about $\Delta\Phi = \pm\pi$;
2. two regions of binaries librating about $\Delta\Phi = \pm\pi$ separated by a third region of circulating binaries;
3. three different regions, where binaries librate about $\Delta\Phi = 0$, circulate and librate about $\Delta\Phi = \pm\pi$.

This distinction is performed by `phase_xi`. A useful tool is provided in `phase_checker`, which ensures that the output of `phase_xi` satisfies the constraints of Sec. III C.

IV. GRAVITATIONAL-WAVE DRIVEN INSPIRAL

In this section we illustrate how to use PRECESSION to compute BH inspirals. We provide a standard integrator of the orbit-averaged PN equations (Sec. IV A) and a framework to evolve binaries using our innovative precession-averaged approach (Sec. IV B). A key ingredient is the statistical resampling of the precessional phase, which is illustrated in Sec. IV C. Finally, we present a new hybrid approach where precessional cycles are tracked only during the last part of the inspiral (Sec. IV D).

A. Orbit-averaged evolutions

GW emission dissipates energy and angular momentum, thus decreasing the binary separation. Following the seminal studies of Apostolatos et al. [2] and Kidder [3], the PN equations of motion for precessing systems have historically been studied averaging over the orbital motion [73, 76, 83–86], which exploits the inequalities $t_{\rm orb} \ll t_{\rm pre}$ and $t_{\rm orb} \ll t_{\rm RR}$. We provide a numerical integrator for the following set of orbit-averaged PN equations:

$$\frac{d\mathbf{S_1}}{dt} = \mathbf{\Omega_1}\times\mathbf{S_1}, \qquad \frac{d\mathbf{S_2}}{dt} = \mathbf{\Omega_2}\times\mathbf{S_2}, \qquad \frac{d\hat{\mathbf{L}}}{dt} = -\frac{v}{\eta M^2}\frac{d}{dt}(\mathbf{S_1}+\mathbf{S_2}); \tag{24}$$

$$M\mathbf{\Omega_1} = \eta v^5\left(2+\frac{3q}{2}\right)\hat{\mathbf{L}} + \frac{v^6}{2M^2}\left[\mathbf{S_2} - 3\left(\hat{\mathbf{L}}\cdot\mathbf{S_2}\right)\hat{\mathbf{L}} - 3q\left(\hat{\mathbf{L}}\cdot\mathbf{S_1}\right)\hat{\mathbf{L}}\right]; \tag{25}$$

$$M\mathbf{\Omega_2} = \eta v^5\left(2+\frac{3}{2q}\right)\hat{\mathbf{L}} + \frac{v^6}{2M^2}\left[\mathbf{S_1} - 3\left(\hat{\mathbf{L}}\cdot\mathbf{S_1}\right)\hat{\mathbf{L}} - \frac{3}{q}\left(\hat{\mathbf{L}}\cdot\mathbf{S_2}\right)\hat{\mathbf{L}}\right]; \tag{26}$$

$$\begin{aligned}\frac{dv}{dt} = \frac{32}{5}\frac{\eta}{M}v^9\Bigg\{1 - v^2\frac{743+924\eta}{336} + v^3\left[4\pi - \sum_{i=1,2}\chi_i(\hat{\mathbf{S}}_\mathbf{i}\cdot\hat{\mathbf{L}})\left(\frac{113}{12}\frac{m_i^2}{M^2}+\frac{25}{4}\eta\right)\right] + v^4\left[\frac{34103}{18144} + \frac{13661}{2016}\eta + \frac{59}{18}\eta^2\right.\\ \left. + \frac{\eta\chi_1\chi_2}{48}\left(721(\hat{\mathbf{S}_1}\cdot\hat{\mathbf{L}})(\hat{\mathbf{S}_2}\cdot\hat{\mathbf{L}}) - 247(\hat{\mathbf{S}_1}\cdot\hat{\mathbf{S}_2})\right) + \frac{1}{96}\sum_{i=1,2}\left(\frac{m_i\chi_i}{M}\right)^2\left(719(\hat{\mathbf{S}}_\mathbf{i}\cdot\hat{\mathbf{L}})^2 - 233\right)\right] - v^5\pi\frac{4159+15876\eta}{672}\\ + v^6\left[\frac{16447322263}{139708800} + \frac{16}{3}\pi^2 - \frac{1712}{105}\left(\gamma_E + \ln 4v\right) + \left(\frac{451}{48}\pi^2 - \frac{56198689}{217728}\right)\eta + \frac{541}{896}\eta^2 - \frac{5605}{2592}\eta^3\right]\\ + v^7\pi\left[-\frac{4415}{4032} + \frac{358675}{6048}\eta + \frac{91495}{1512}\eta^2\right] + O(v^8)\Bigg\};\end{aligned} \tag{27}$$

where $v = \sqrt{M/r}$ is the orbital velocity and $\gamma_E \simeq 0.577$ is Euler's constant. The spin-precession equations (24)-(26) are accurate up to 2PN; corrections to the radiation-reaction equation (27) are included up 3.5PN (2PN) for (non-)spinning terms [2–4, 68, 73, 85–92]. Higher-order PN corrections to spin precession [93–95] and radiation reaction [96, 97] are not implemented in the current version of PRECESSION. The importance of such additional corrections on the conservation of ξ and their quantitative effect at small separations are still unclear and surely merit further investigation.

Orbit-averaged inspirals require the integration of nine coupled ordinary differential equations (ODEs) for the components of $\mathbf{L}$, $\mathbf{S_1}$ and $\mathbf{S_2}$. Although the time t at a given separation r is crucial to calculate the emitted GW signal, it is not relevant for most astrophysical purposes, where only the evolution of the spin orientations is needed. For this reason, PRECESSION performs PN integrations using the separation r as independent variable. In practice, we integrate $d\mathcal{L}/dr = d\mathcal{L}/dt \times (dv/dt)^{-1} \times 1/2\sqrt{rM}$, where $\mathcal{L}$ is any of the components of $\mathbf{L}$, $\mathbf{S_1}$ and $\mathbf{S_2}$. Integrations are performed using the `lsoda` algorithm [98] implemented in `scipy.integrate.odeint`. `lsoda` combines adaptive nonstiff and stiff methods and monitors the ODE integrations to switch between the two as needed.

We provide three convenient wrappers of the orbit-averaged PN integrator, which differ in their input and output parameters.

1. `orbit_averaged` evolves the relative orientation of the three momenta given in terms of (ξ, J, S). The initial configurations must be compatible with the constraints presented in Sec. III C.

2. `orbit_angles` evolves BH binary configurations specified by the angles $(\theta_1, \theta_2, \Delta\Phi)$.

3. `orbit_vectors` tracks the evolution of the nine components of $\mathbf{L}$ of $\mathbf{S_1}$ and $\mathbf{S_2}$ in an inertial frame.

In the first two cases, the integration is carried out in a reference frame $(\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}})$ defined by $\mathbf{J}\cdot\hat{\mathbf{x}} = \mathbf{J}\cdot\hat{\mathbf{y}} = \mathbf{L}\cdot\hat{\mathbf{y}} = 0$ at the initial separation; generic configurations can be projected to this frame using `Jframe_projection`. In the third case, the integration frame is specified by the input parameters. Examples are shown in Sec. VI D. Parallelization is implemented in all wrappers to evolve distributions of BH binaries on multiple cores (cf. Sec. VI F). If needed, the right-hand side of Eqs. (25)-(27) can be accessed explicitly calling `orbav_eqs`.

B. Precession-averaged evolutions

Refs. [6, 7] introduced an alternative way to evolve spinning BH binaries, which explicitly exploits the timescale hierarchy $t_{\rm pre} \ll t_{\rm RR}$. The three parameters (ξ, J, S) describing the relative orientations of the BH spins naturally accommodate the timescales of the problem:

- ξ is conserved on both $t_{\rm pre}$ and $t_{\rm RR}$;
- J is conserved on $t_{\rm pre}$ but varies on $t_{\rm RR}$;
- S varies on both $t_{\rm pre}$ and $t_{\rm RR}$.

The oscillations of S on $t_{\rm pre}$ can be averaged over to study the binary evolution on times $t \sim t_{\rm RR}$. The secular variation of J on $t_{\rm RR}$ is given at 1PN by

$$\frac{dJ}{dr} = \frac{1}{4rJ}\left(J^2 + L^2 - \frac{\int_{S_-}^{S_+} S^2 |dS/dt|^{-1} dS}{\int_{S_-}^{S_+} |dS/dt|^{-1} dS}\right) . \quad (28)$$

This approach reduces the PN evolution of a BH binary to a single ODE. The price paid to achieve this simplification is the loss of information on the evolution of S (cf. Sec. IV C below).

The integration domain of Eq. (28) can be extended to arbitrarily large separations using auxiliary variables

$$\kappa = \frac{J^2 - L^2}{2L} , \qquad u = \frac{1}{2L} , \quad (29)$$

such that Eq. (28) reduces to

$$\frac{d\kappa}{du} = \frac{\int_{S_-}^{S_+} S^2 |dS/dt|^{-1} dS}{\int_{S_-}^{S_+} |dS/dt|^{-1} dS} , \quad (30)$$

which can be integrated from/to $u = 0$ ($r/M = \infty$). While $J \sim L \propto \sqrt{r}$ diverges in the large separation limit, the asymptotic value of κ

$$\kappa_\infty = \lim_{r/M\to\infty} \kappa = \lim_{r/M\to\infty} (\mathbf{S_1} + \mathbf{S_2})\cdot\hat{\mathbf{L}} \quad (31)$$

converges and becomes equivalent to the projection of the total spin along the orbital angular momentum. κ_∞ is therefore bounded by

$$-(S_1 + S_2) \leq \kappa_\infty \leq S_1 + S_2, \quad (32)$$

as given by `kappainf_lim`. BH binary configurations at infinitely large separation are specified by pairs (ξ, κ_∞) satisfying Eqs. (11) and (32), see Sec. VI A. The allowed range of these two parameters can be computed using `kappainf_allowed` and `xiinf_allowed`. θ_1 and θ_2 asymptote to finite values at large separation, and can be expressed in terms of ξ and κ_∞:

$$\cos\theta_{1\infty} \equiv \lim_{r/M\to\infty} \cos\theta_1 = \frac{\kappa_\infty(1+q^{-1}) - M^2\xi}{S_1(q^{-1} - q)} , \quad (33)$$

$$\cos\theta_{2\infty} \equiv \lim_{r/M\to\infty} \cos\theta_2 = \frac{M^2\xi - \kappa_\infty(1+q)}{S_2(q^{-1} - q)} . \quad (34)$$

Transformations between (ξ, κ_∞) and $(\theta_{1\infty}, \theta_{2\infty})$ are implemented in `thetas_inf` and `from_the_angles_inf`.

PRECESSION provides three different wrappers to integrate Eqs. (28) and (30):

1. `evolve_J` evolves the binary between two finite separations r_i and r_f. The initial condition $J(r_i)$ must satisfy the geometrical constraints of Sec. III C.

2. `evolve_J_infinity` integrates Eq. (30) from $r/M = \infty$ ($u = 0$) down to some final separation r_f. The initial configuration has to be specified in terms of κ_∞.

3. `evolve_J_backwards` evolves a binary specified at some separation r_i back to past infinity and returns its asymptotic condition κ_∞.

Practical examples are provided in Sec. VI D. Integrations are performed using the `lsoda` algorithm [98] wrapped in `scipy.integrate.odeint`. Parallelization is implemented to run arrays of binaries simultaneously (cf. Sec. VI F). The right-hand side of Eqs. (28) and (30) can be evaluated directly using `dJdr` and `dkappadu`.

When performing precession-averaged evolutions, we recommend avoiding binary configurations very close to the limits reported in Eqs. (11)-(15). Numerical errors arising from the integration of Eq. (28) may push some of the parameters out of their range of validity, which

prevents any further evolution. PRECESSION is rather solid with respect to such errors: tolerances as small as $\Delta J/M^2 \sim \Delta\xi \sim 10^{-6}$ from the limits reported in Eqs. (11)-(15) are typically sufficient to ensure smooth integrations.

C. Phase resampling and binary transfer

Precession-averaged integrations do not track the evolution of the precessional phase. This is a well justified approach for most astrophysical applications. Interactions with the astrophysical environment determine the spin orientation at large separation where GW emission is inefficient to drive the dynamics [34, 50, 51, 53, 64, 99–103]. The inequality $t_{\rm pre} \ll t_{\rm RR}$ implies that BH binaries undergo a very large number of precession cycles before entering the GW-driven regime, such that the information of the initial phase is lost in practice.

An estimate of the magnitude of the total spin S is nonetheless available at a statistical level from the dynamics on the shorter times $t \sim t_{\rm pre}$. The probability of finding a binary with some total spin magnitude S is proportional to dt/dS of Eq. (17). We sample the probability distribution $P(S) = 2|dS/dt|^{-1}/\tau$ (with $S \in [S_-, S_+]$) using the cumulative distribution method (e.g. [104]), which is suitable to handle integrable singularities (cf. Eq. 20). We first select a random number $\epsilon \in [0, 1]$ and then solve the integral equation

$$\frac{2}{\tau}\int_{S_-}^{S} \frac{dS'}{|dS'/dt|} = \epsilon \tag{35}$$

for $S \in [S_-, S_+]$. The algorithm is implemented in `samplingS` and tested in Sec. VI C below.

Phase resampling is essential to transfer the spin orientations of BH binaries from large separation where they form down to the regime close to merger. The complete procedure is implemented in `evolve_angles`, and can be summarized as follows.

1. We specify a binary with mass ratio q, spin magnitudes S_1, S_2 and spin orientations $(\theta_1, \theta_2, \Delta\Phi)$ at some initial separation r_i.

2. We convert the initial configuration to (ξ, J, S) but only consider (ξ, J), thus explicitly losing memory of S.

3. The configuration (ξ, J) is evolved down to some final separation r_f integrating Eq. (28) for J (ξ stays constant).

4. Given the final configuration (ξ, J) at r_f, we randomly extract a value S from a distribution weighted by $|dS/dt|^{-1}$.

5. The final set of parameters (ξ, J, S) is converted back to $(\theta_1, \theta_2, \Delta\Phi)$. The sign of $\Delta\Phi$ is randomly chosen.

This procedure allows for direct comparison between orbit-averaged and precession-averaged evolutions. Such a comparison is carried out in Sec. VI E as a test of the code. Tests performed on distributions of binaries have been reported by [7], where precession-averaged binary transfers have been found to be in excellent statistical agreement with orbit-averaged evolutions. Discrepancies between the two approaches become relevant only at $r \sim 10M$, where $t_{\rm pre}$ becomes comparable to $t_{\rm RR}$. However, the entire PN approach loses accuracy at such small separations [77, 105, 106] and the binary evolution can be followed faithfully only using numerical-relativity simulations.

Neglecting and resampling the precessional phase lead to a substantial computational speed up. A concrete example is provided in Sec. VI F: even starting at moderate separation $\sim 10^4 M$, precession-averaged integrations are faster by about a factor ~ 70 when compared to orbit-averaged evolutions[4]. Orbit-averaged integrations become impractical at separations significantly larger than $\sim 10^4 M$, while precession-averaged evolutions can be carried out to/from infinitely large separation.

D. Hybrid evolutions

Although optimal for statistical studies, phase resampling may be inaccurate in situations where individual precession cycles need to be resolved. PRECESSION can perform hybrid PN integrations combining the two approaches in `hybrid`:

1. A precession-averaged integration is used at large separations, down to a certain separation threshold r_t.

2. The precessional phase is extracted at r_t by resampling the total spin magnitude S.

3. This binary configuration at r_t is used to initialize an orbit-averaged integration to resolve individual precession cycles at separations $r < r_t$.

The transition radius r_t may correspond, for instance, to a typical separation where the emitted GW frequency $f_t = \sqrt{M/\pi^2 r_t^3}$ enters the lower end of the sensitivity window of a specific detector. For convenience, we provide utilities to convert binary separation and emitted GW frequency in `rtof` and `ftor`.

[4] Precession-averaged evolutions may occasionally stall and take longer to run. This is due to a wrong initial guess of the integration step attempted by `lsoda` and can be cured increasing the `h0` optional parameter of `scipy.integrate.odeint`. With the current default option, stalling happens roughly once every million inspirals.

V. BLACK-HOLE REMNANTS

PRECESSION implements numerical-relativity fitting formulae to estimate final mass (Sec. V A), spin (Sec. V B) and recoil (Sec. V C) of BHs following binary mergers. The importance of spin precession in estimating these properties is stressed in Sec. V D.

The fitting formulae are typically written down using the following weighted combinations of the BH spins

$$\mathbf{\Delta} = \frac{q\chi_2\hat{\mathbf{S}}_\mathbf{2} - \chi_1\hat{\mathbf{S}}_\mathbf{1}}{1+q}\,, \qquad \tilde{\boldsymbol{\chi}} = \frac{q^2\chi_2\hat{\mathbf{S}}_\mathbf{2} + \chi_1\hat{\mathbf{S}}_\mathbf{1}}{(1+q)^2}, \tag{36}$$

and their projections parallel/perpendicular to the orbital angular momentum: $\tilde{\chi}_\parallel = \tilde{\boldsymbol{\chi}}\cdot\hat{\mathbf{L}}$, $\tilde{\chi}_\perp = |\tilde{\boldsymbol{\chi}}\times\hat{\mathbf{L}}|$, $\Delta_\parallel = \mathbf{\Delta}\cdot\hat{\mathbf{L}}$, $\Delta_\perp = |\mathbf{\Delta}\times\hat{\mathbf{L}}|$.

A. Final mass

The energy radiated in GWs during the inspiral and merger of a BH binary decreases the mass of the BH remnant M_f below the binary's total mass M. Estimates of M_f can be computed analytically in the test-particle limit $q\to 0$ [107] and numerically in the strong-field regime $q\simeq 1$ [74, 108, 109]. An interpolation between these two regimes is given in [110] and reads

$$\begin{aligned}\frac{M_f}{M} &= 1 - \eta(1+4\eta)\,(1-E_{\rm ISCO}) \\ &\quad - 16\eta^2\left[p_0 + 4p_1\tilde{\chi}_\parallel\left(\tilde{\chi}_\parallel+1\right)\right]\,.\end{aligned} \tag{37}$$

Here $E_{\rm ISCO}$ is the energy per unit mass of an effective particle of spin $\tilde{\boldsymbol{\chi}}$ at the innermost stable circular orbit [111]:

$$E_{\rm ISCO} = \sqrt{1-\frac{2}{3r_{\rm ISCO}}}\,, \tag{38}$$

$$r_{\rm ISCO} = 3 + Z_2 - {\rm sign}(\tilde{\chi}_\parallel)\sqrt{(3-Z_1)(3+Z_1+2Z_2)}\,, \tag{39}$$

$$Z_1 = 1 + \left(1-\tilde{\chi}_\parallel^2\right)^{1/3}\left[\left(1+\tilde{\chi}_\parallel\right)^{1/3} + \left(1-\tilde{\chi}_\parallel\right)^{1/3}\right]\,, \tag{40}$$

$$Z_2 = \sqrt{3\tilde{\chi}_\parallel^2 + Z_1^2}. \tag{41}$$

The parameters $p_0 = 0.04827$ and $p_1 = 0.01707$ have been obtained by [110] fitting 186 numerical-relativity simulations from various groups. M_f can be computed calling `finalmass`.

B. Final spin

A convenient expression for the spin $S_f = M_f^2\chi_f$ of the BH remnant is given in [112], where test-particle results [107, 113] and numerical-relativity simulations [74, 109, 114] are interpolated. Their expression for the dimensionless spin χ_f is implemented in `finalspin` and reads

$$\chi_f = \min\left(1, \left|\tilde{\boldsymbol{\chi}} + \frac{q}{(1+q)^2}\ell\hat{\mathbf{L}}\right|\right)\,, \tag{42}$$

$$\begin{aligned}\ell &= 2\sqrt{3} + t_2\eta + t_3\eta^2 + s_4\frac{(1+q)^4}{(1+q^2)^2}\tilde{\chi}^2 \\ &\quad + (s_5\eta + t_0 + 2)\frac{(1+q)^2}{1+q^2}\tilde{\chi}_\parallel\,,\end{aligned} \tag{43}$$

with $t_0 = -2.8904$, $t_2 = -3.51712$, $t_3 = 2.5763$, $s_4 = -0.1229$ and $s_5 = 0.4537$.

Various alternative prescriptions for the final spin have been compared in [39], where the critical importance of accounting for PN spin precession in estimating χ_f is demonstrated (see also the discussion by [112] on this point).

C. Black-hole recoil

If GWs are emitted anisotropically during inspiral and merger, linear momentum is dissipated in a preferential direction and the center of mass recoils in the opposite direction. BH recoil (or kick) velocities v_k can be as large as ~ 5000 km/s, which exceeds the escape velocities of the most massive galaxies [115]. Kicks are generated by asymmetries in either the masses or the spins of the two merging BHs. The mass asymmetry contribution to the kick velocity v_m lies in the orbital plane, while the spin contribution has components $v_{s\parallel}$ and $v_{s\perp}$ directed parallel and perpendicular to the orbital angular momentum. The magnitude of the kick velocity v_k can be modeled as [116]

$$v_k = \sqrt{v_m^2 + 2v_m v_{s\perp}\cos\zeta + v_{s\perp}^2 + v_{s\parallel}^2}\,, \tag{44}$$

where ζ is the angle between the mass term and the orbital-plane spin term. We have implemented the following expressions for v_m, $v_{s\perp}$ and $v_{s\parallel}$:

$$v_m = A\eta^2\frac{1-q}{1+q}(1+B\eta)\,, \tag{45}$$

$$v_{s\perp} = H\eta^2\Delta_\parallel\,, \tag{46}$$

$$\begin{aligned}v_{s\parallel} &= 16\eta^2[\Delta_\perp(V_{11} + 2V_A\tilde{\chi}_\parallel + 4V_B\tilde{\chi}_\parallel^2 + 8V_C\tilde{\chi}_\parallel^3) \\ &\quad + 2\tilde{\chi}_\perp\Delta_\parallel(C_2 + 2C_3\tilde{\chi}_\parallel)]\cos\Theta\,,\end{aligned} \tag{47}$$

where the coefficients are extracted from numerical-relativity simulations: $A = 1.2\times10^4$ km/s, $B = -0.93$ [117], $H = 6.9\times10^3$ km/s [118], $V_{11} = 3677.76$ km/s, $V_A = 2481.21$ km/s, $V_B = 1792.45$ km/s, $V_C = 1506.52$ km/s [119], $C_2 = 1140$ km/s, $C_3 = 2481$ km/s [120], $\zeta = 145°$ [118]. The main contribution to v_k comes from the term proportional to V_{11} in Eq. (47). This effect, known as "superkick", enters v_k weighted by $\Delta_\perp$ and it is dominant if binaries merge with $\theta_i\sim\pi/2$ and $\Delta\Phi\sim\pi$

[116, 121]. The additional corrections $V_{A,B,C}$ ($C_{2,3}$) are known as "hangup-kicks" ("cross-kicks") and increase v_k for moderate misalignments $\theta_i \sim 50^\circ$ [120, 122]. The additional parameter Θ is the angle between the direction of $\boldsymbol{\Delta} \times \hat{\mathbf{L}}$ and the infall direction of the two holes "at merger", offset by $\sim 200^\circ$ [123, 124]. In practice, Θ depends on the initial separation of the BH binary in each numerical-relativity simulation. Following previous studies [49, 119, 125], PRECESSION deals with this dependency assuming Θ to be uniformly distributed in $[0, \pi]$. Possible PN effects on the probability distribution of Θ are not taken into account.

Eq. (44) can be evaluated using `finalkick` and predicts a maximum kick velocity $v_k \sim 0.017c \sim 5000$ km/s.

D. Importance of spin precession

Spin precession plays a crucial role in determining the properties of the BH remnant. The fitting formulae here presented should only be applied at separations $r \lesssim 10M$ comparable to the initial conditions of the numerical-relativity simulations used in their calibration. The PN inspiral before merger profoundly modifies the spin orientations and therefore the estimated properties of the final BH. Ref. [39] showed that PN spin precession introduces a fundamental uncertainty in predicting the final spin because χ_f depends on the precessional phase at merger, which is only available at a statistical level. This point is even more crucial for kick predictions. Large kicks are expected to be less (more) likely if binaries merge with $\Delta\Phi \sim 0$ ($\sim \pi$) [116, 121] and, consequently, phase transitions towards the librating morphologies during the early inspiral substantially suppress (enhance) the recoil [40, 125]. Ref. [7] found that binary morphologies close to merger are closely related to the spin configurations at large separations, which opens up the possibility of exploiting future BH kick measurements to constrain the astrophysical processes behind BH binary formation and evolution [36, 47–49, 52, 102, 126].

The expressions for final mass, spin and recoil currently implemented in PRECESSION are the same already presented in [47]; other recent findings (e.g. [74, 127–129]) will be implemented in future versions of the code.

VI. EXAMPLES

This section contains several practical examples to use PRECESSION. All tests presented here are available in the PYTHON submodule `precession.test` which has to be loaded explicitly with the command:

```
import precession.test
```

The source code of the example routines are reported in Figs. 2-7. The outcome of their executions are presented as screen outputs or graphical plots in Figs. 8-13. Each example is described in a dedicated subsection: Sec. VI A shows how to select consistent BH binary configurations and study their dynamics on $t_{\rm pre}$; in Sec. VI B, we study the precessional cycles of three BH binaries and classify their spin morphologies; in Sec. VI C, we test our algorithm to resample the precessional phase; Sec. VI D shows how to compute PN inspirals and evaluate numerical-relativity fitting formulae to estimate the properties of the post-merger BH; finally, in Sec. VI E and VI F we compare binary dynamics and computational speed of orbit-averaged and precession-averaged integrations.

A. Selection of consistent parameters

The function `test.parameter_selection` illustrates how to select consistent parameters and characterize the binary dynamics on the precession timescale. The test is executed with

```
precession.test.parameter_selection()
```

The source code and the screen output are reported in Figs. 2 and 8 respectively.

We first show how to select values of (ξ, J, S) that satisfy the geometrical constraints described in Sec. III C, and how to convert these values to $(\theta_1, \theta_2, \Delta\Phi)$. Secondly, we compute several quantities that characterize BH spin precession: the angles θ_i corresponding to the spin-orbit resonances, the precessional period τ, the total precession rate α and the spin morphology. Finally, we illustrate how to select consistent parameters at infinitely large separation $r/M \to \infty$ (cf. Sec. IV B).

B. Evolutions of the spin angles on a precession cycle

The function `test.spin_angles` provides an example to study the binary evolution over one single precession cycle. The test is executed with

```
precession.test.spin_angles()
```

The source code is reported in Fig. 3; the resulting plot is shown in Fig. 9.

The separation r and the magnitude of the total angular momentum J are approximately constant on times $t \sim t_{\rm pre}$. Combined with the conservation of ξ, this implies that the precessional dynamics can be parametrized using a single parameter. We first parametrize the precession cycles using the magnitude of the total spin S. Time evolutions are then obtained by integrating dS/dt according to Eq. (19). The magnitude S undergoes a full oscillation between two values S_- and S_+ in a time τ [cf. Eq. (21)], which defines the precession period. As shown in Fig. 9, the evolution of the tilt angles θ_i is qualitatively similar for all binaries. On the other hand, three different situations are possible for $\Delta\Phi$ and exemplify the notion of precessional morphology. As already pointed out in Sec. III B, the sign of $\Delta\Phi$ must be specified by the user:

one has $\Delta\Phi \leq 0$ ($\Delta\Phi \geq 0$) in the first (second) half of the precession cycle where S increases (decreases).

C. Sampling of the precessional phase

The routine `test.phase_sampling` tests our procedure to statistically sample values of S weighted by $|dS/dt|^{-1}$ (cf. Sec. IV C). The test is executed with

```
precession.test.phase_sampling()
```

The source code is reported in Fig. 4; the resulting plot is shown in Fig. 10.

After selecting a BH binary configuration $(q, \chi_1, \chi_2, r, J, \xi)$, we extract multiple values of $S \in [S_-, S_+]$ using `samplingS`. The obtained distribution is normalized, binned, and compared with the continuum limit $P(S) = 2|dS/dt|^{-1}/\tau$. As a consistency check, we also convert our sample to $t(S)$ using Eq. (19) and verify that these values are distributed uniformly. This example also demonstrates that the singularities of $P(S)$ at $S_\pm$ [cf. Eq. (20)] are integrable and result in a smooth probability distribution of t.

D. Wrappers of the PN integrators

The example `test.PNwrappers` shows how to perform PN inspirals using the ODE integrators implemented in PRECESSION. The test is executed with

```
precession.test.PNwrappers()
```

The source code and the screen output are reported in Figs. 5 and 11, respectively.

We first specify a BH binary at some initial separation r_i by providing values of the angles $(\theta_1, \theta_2, \Delta\Phi)$, which are then converted to (ξ, J, S), cf. Sec. III B. This system is first evolved down to a final separation $r_f < r_i$ integrating the orbit-averaged PN equations of motion (25)-(27). The integration is performed using the three wrappers presented in Sec. IV A to extract the final configuration in terms of $(\theta_1, \theta_2, \Delta\Phi)$, (ξ, J, S), and the nine components of $\mathbf{L}$, $\mathbf{S_1}$, $\mathbf{S_2}$. The same evolution is then performed using the precession-averaged approach outlined in Sec. IV B. In contrast to orbit-averaged integrations, ξ is not evolved explicitly and it is assumed to be constant. The final value of J is obtained by integrating Eq. (28). The evolution of S is not tracked explicitly, but can be resampled (cf. Sec. IV C) to obtain a statistical estimate of the angles $(\theta_1, \theta_2, \Delta\Phi)$ at r_f. We then show how to perform integrations to/from $r/M \to \infty$, where the projection of the total spin κ_∞ is asymptotically constant. Finally, we evolve the same BH binary using a hybrid approach, stitching together precession-averaged and orbit-averaged integrations at some separation r_t. We complete this exercise with the evaluation of the numerical-relativity fitting formulae to estimate the properties of the post-merger BH remnant (Sec. V). Formulae are applied at r_f, *after* the PN evolution.

E. Comparison between orbit-averaged and precession-averaged integrations

The example `test.compare_evolutions` compares a single PN evolution performed using orbit-averaged and precession-averaged integrations. The test is executed with

```
precession.test.compare_evolutions()
```

The source code is reported in Fig. 6; the resulting plot is shown in Fig. 12.

Conservation of the effective spin ξ on the precessional time [73, 79] is a crucial assumption underlying our precession-averaged approach. On the other hand, orbit-averaged integrations confirm this feature as a by-product. We detect extremely small deviations $\Delta\xi/\xi \sim 10^{-11}$ between the two approaches (cf. top panel of Fig. 12), which fully corroborates our assumption, at least at the PN order we implemented. Variations of ξ due to additional PN corrections [93–95] still need to be explored. Note that ξ is *not* conserved on the orbital timescale (only on $t_{\rm pre}$ and $t_{\rm RR}$), but those variations are not captured by either of our methods. The evolution of J is also very accurate, with deviations of the order of $\Delta J/J \sim 10^{-3}$ during the entire integration (Fig. 12, middle panel). The precession-average approach gradually loses accuracy at small separations, where the precession time $t_{\rm pre}$ becomes comparable to the inspiral time $t_{\rm RR}$. Precession-averaged integrations require a resampling of the precessional phase S at each output separation. Resampled values are in excellent *statistical* agreement with the orbit-averaged result (lower panel of Fig. 12). The envelope of the orbit-averaged evolution of S is well described by the $S_\pm$ curves given by $\xi_\pm(S) = \xi$, cf. Eq. (16).

F. Parallel computation and timing

Our last example, `test.timing`, compares the computational efficiency of the PN integrators implemented in PRECESSION. The test is executed with

```
precession.test.timing()
```

The source code and the screen output are reported in Figs. 7 and 13, respectively.

We compute the CPU time needed to evolve a sample of $N = 100$ BH binaries from $r_i = 10^4 M$ to $r_f = 10M$ using orbit-averaged and precession-averaged integrations. In particular, we time the orbit-averaged integrator wrapped inside `orbit_angles` (cf. Sec. IV A) against the precession-averaged evolution implemented in `evolve_angles` (cf. Sec. IV B). The latter includes both the numerical integration of Eq. (28) and a final resampling of the magnitude S. To better illustrate the parallel implementation of the integrators, we perform the same computation twice: in the first iteration, integrations are performed in parallel on all the available cores (default); in the second iteration, we enforce a strictly serial execution. On average, a single BH inspiral takes ~ 3 minutes

($\sim$ 3 seconds) when evolved using orbit- (precession-) averaged integrations. The computational performances obtained here are in good agreement with [7], where the dependence of the CPU time on the initial separation r_i is also studied (see their Fig. 9).

VII. CONCLUSIONS

We have presented design and usage of the numerical open-source code PRECESSION. Our code provides various numerical tools to study the precessional dynamics of BH binaries, evolve BH binaries along their GW-driven inspirals and estimate the properties of the single BHs resulting from binary mergers. PRECESSION is distributed in the form of a PYTHON module to combine flexibility, ease-of-use and numerical efficiency. The code can be straightforwardly installed from the PYTHON Package Index through `pip`, and it is distributed under version control at `github.com/dgerosa/precession`. Extensive documentation is regularly maintained at `dgerosa.github.io/precession`. Further information is available at `davidegerosa.com/precession`.

PRECESSION is under active development and several features will be added in future versions. Possible extensions include (i) generalization to eccentric orbits, (ii) explicit treatment of single-spin and non-spinning binaries, (iii) reparametrization of the dynamics in the equal-mass limit [75], (iv) implementation of the latest fitting formulae to numerical-relativity simulations, (v) addition of higher-order PN corrections, and (vi) inclusion of numerical tools to study the resonant configurations $\alpha = 2\pi n$ [82]. On the computational side, PRECESSION will be ported to PYTHON 3, and its parallel computing features further refined. Additional computational speed-up could be achieved using static compilers such as CYTHON [130]. Compatibility and/or integration with the LIGO Algorithm Library[5] software is also an important future development.

The numerical tools described in this paper facilitate the implementation of spinning BH binary inspirals in a variety of astrophysical studies, ranging from population synthesis models to galaxy merger trees. Moreover, PRECESSION provides flexible tools to interpret GW observations and numerical-relativity simulations of BH binaries in light of multi-timescale PN techniques. As merging BH binaries have entered the realm of observations, we hope that our numerical efforts –here made available to the scientific community– will help understanding these fascinating physical systems straddling the boundaries between fundamental physics and astronomy.

ACKNOWLEDGMENTS

We are grateful to Ulrich Sperhake, Emanuele Berti, Richard O'Shaughnessy, Alberto Sesana, Daniele Trifiró, Antoine Klein, Tyson Littenberg, Jakub Vosmera, Xinyu Zhao, Will Farr, Enrico Barausse and Guillame Faye for several fruitful discussions. This work was inspired by [131]. D.G. is supported by the UK STFC and the Isaac Newton Studentship of the University of Cambridge. Partial support is also acknowledged from the Royal Astronomical Society, Darwin College of the University of Cambridge, the Cambridge Philosophical Society, the H2020 ERC Consolidator Grant No. MaGRaTh–646597, the H2020-MSCA-RISE-2015 Grant No. StronGrHEP-690904, the STFC Consolidator Grant No. ST/L000636/1, the SDSC Comet and TACC Stampede clusters through NSF-XSEDE Award Nos. PHY-090003, the Cambridge High Performance Computing Service Supercomputer Darwin using Strategic Research Infrastructure Funding from the HEFCE and the STFC, and DiRAC's Cosmos Shared Memory system through BIS Grant No. ST/J005673/1 and STFC Grant Nos. ST/H008586/1, ST/K00333X/1. M. K. is supported by Alfred P. Sloan Foundation Grant No. FG-2015-65299. This work was made possible by the open-source programming language PYTHON [58] and the related tools NUMPY [59], SCIPY [60], MATPLOTLIB [61], PARMAP [62] and PDOC [67]. Version-control distribution through GIT and GITHUB is also acknowledged.

[1] R. P. Kerr, PRL **11**, 237 (1963).
[2] T. A. Apostolatos, C. Cutler, G. J. Sussman, and K. S. Thorne, PRD **49**, 6274 (1994).
[3] L. E. Kidder, PRD **52**, 821 (1995), gr-qc/9506022.
[4] P. C. Peters and J. Mathews, Physical Review **131**, 435 (1963).
[5] E. Poisson and C. M. Will, *Gravity* (Cambridge University Press, 2014).
[6] M. Kesden, D. Gerosa, R. O'Shaughnessy, E. Berti, and U. Sperhake, PRL **114**, 081103 (2015), arXiv:1411.0674 [gr-qc].
[7] D. Gerosa, M. Kesden, U. Sperhake, E. Berti, and R. O'Shaughnessy, PRD **92**, 064016 (2015), arXiv:1506.03492 [gr-qc].
[8] F. Pretorius, (2007), arXiv:0710.1338 [gr-qc].
[9] E. E. Salpeter, ApJ **140**, 796 (1964).
[10] Y. B. Zel'dovich, Soviet Physics Doklady **9**, 195 (1964).
[11] M. Schmidt, Nature **197**, 1040 (1963).
[12] B. L. Webster and P. Murdin, Nature **235**, 37 (1972).
[13] C. T. Bolton, Nature **235**, 271 (1972).
[14] J. Casares and P. G. Jonker, Space Sci. Rev. **183**, 223 (2014), arXiv:1311.5118 [astro-ph.HE].
[15] J. Kormendy and D. Richstone, ARA&A **33**, 581 (1995).
[16] C. S. Reynolds, Classical and Quantum Gravity **30**, 244004 (2013), arXiv:1307.3246 [astro-ph.HE].

[5] LAL, `www.lsc-group.phys.uwm.edu/lal`.

[17] M. C. Miller and J. M. Miller, Phys. Rep. **548**, 1 (2015), arXiv:1408.4145 [astro-ph.HE].
[18] K. A. Postnov and L. R. Yungelson, Living Reviews in Relativity **17** (2014), arXiv:1403.4754 [astro-ph.HE].
[19] M. J. Benacquista and J. M. B. Downing, Living Reviews in Relativity **16** (2013), arXiv:1110.4423 [astro-ph.SR].
[20] S. D. M. White and M. J. Rees, MNRAS **183**, 341 (1978).
[21] M. C. Begelman, R. D. Blandford, and M. J. Rees, Nature **287**, 307 (1980).
[22] C. Rodriguez, G. B. Taylor, R. T. Zavala, A. B. Peck, L. K. Pollack, and R. W. Romani, ApJ **646**, 49 (2006), astro-ph/0604042.
[23] B. P. Abbott *et al.*, PRL **116**, 061102 (2016), arXiv:1602.03837 [gr-qc].
[24] B. P. Abbott *et al.*, (2016), arXiv:1602.03838 [gr-qc].
[25] P. A. Seoane *et al.*, (2013), arXiv:1305.5720 [astro-ph.CO].
[26] A. Klein, E. Barausse, A. Sesana, A. Petiteau, E. Berti, S. Babak, J. Gair, S. Aoudia, I. Hinder, F. Ohme, and B. Wardell, PRD **93**, 024003 (2016), arXiv:1511.05581 [gr-qc].
[27] F. Jenet *et al.*, (2009), arXiv:0909.1058 [astro-ph.IM].
[28] R. N. Manchester *et al.*, PASA **30**, e017 (2013), arXiv:1210.6130 [astro-ph.IM].
[29] M. Kramer and D. J. Champion, Classical and Quantum Gravity **30**, 224009 (2013).
[30] R. N. Manchester, Classical and Quantum Gravity **30**, 224010 (2013), arXiv:1309.7392 [astro-ph.IM].
[31] Y. Pan, A. Buonanno, A. Taracchini, L. E. Kidder, A. H. Mroué, H. P. Pfeiffer, M. A. Scheel, and B. Szilágyi, PRD **89**, 084006 (2014), arXiv:1307.6232 [gr-qc].
[32] M. Hannam, P. Schmidt, A. Bohé, L. Haegel, S. Husa, F. Ohme, G. Pratten, and M. Pürrer, PRL **113**, 151101 (2014), arXiv:1308.3271 [gr-qc].
[33] K. Chatziioannou, N. Cornish, A. Klein, and N. Yunes, PRD **89**, 104023 (2014), arXiv:1404.3180 [gr-qc].
[34] D. Gerosa, M. Kesden, E. Berti, R. O'Shaughnessy, and U. Sperhake, PRD **87**, 104028 (2013), arXiv:1302.4442 [gr-qc].
[35] K. Chatziioannou, N. Cornish, A. Klein, and N. Yunes, ApJ **798**, L17 (2015), arXiv:1402.3581 [gr-qc].
[36] B. P. Abbott *et al.*, ApJ **818**, L22 (2016), arXiv:1602.03846 [astro-ph.HE].
[37] C. O. Lousto and J. Healy, PRL **114**, 141101 (2015), arXiv:1410.3830 [gr-qc].
[38] C. O. Lousto, J. Healy, and H. Nakano, PRD **93**, 044031 (2016), arXiv:1506.04768 [gr-qc].
[39] M. Kesden, U. Sperhake, and E. Berti, PRD **81**, 084054 (2010), arXiv:1002.2643 [astro-ph.GA].
[40] M. Kesden, U. Sperhake, and E. Berti, ApJ **715**, 1006 (2010), arXiv:1003.4993 [astro-ph.CO].
[41] K. Belczynski, V. Kalogera, F. A. Rasio, R. E. Taam, A. Zezas, T. Bulik, T. J. Maccarone, and N. Ivanova, ApJS **174**, 223-260 (2008), astro-ph/0511811.
[42] M. Volonteri, F. Haardt, and P. Madau, ApJ **582**, 559 (2003), astro-ph/0207276.
[43] E. Barausse, MNRAS **423**, 2533 (2012), arXiv:1201.5888.
[44] V. Springel *et al.*, Nature **435**, 629 (2005), astro-ph/0504097.
[45] M. Vogelsberger, S. Genel, V. Springel, P. Torrey, D. Sijacki, D. Xu, G. Snyder, S. Bird, D. Nelson, and L. Hernquist, Nature **509**, 177 (2014), arXiv:1405.1418.
[46] M. Volonteri, F. Haardt, and K. Gültekin, MNRAS **384**, 1387 (2008), arXiv:0710.5770.
[47] D. Gerosa and A. Sesana, MNRAS **446**, 38 (2015), arXiv:1405.2072.
[48] S. Komossa, Advances in Astronomy **2012**, 364973 (2012), arXiv:1202.1977 [astro-ph.CO].
[49] L. Blecha, D. Sijacki, L. Z. Kelley, P. Torrey, M. Vogelsberger, D. Nelson, V. Springel, G. Snyder, and L. Hernquist, MNRAS **456**, 961 (2016), arXiv:1508.01524.
[50] K. Belczynski, R. E. Taam, E. Rantsiou, and M. van der Sluys, ApJ **682**, 474 (2008), astro-ph/0703131.
[51] T. Fragos, M. Tremmel, E. Rantsiou, and K. Belczynski, ApJ **719**, L79 (2010), arXiv:1001.1107 [astro-ph.HE].
[52] E. Berti and M. Volonteri, ApJ **684**, 822-828 (2008), arXiv:0802.0025.
[53] A. Sesana, E. Barausse, M. Dotti, and E. M. Rossi, ApJ **794**, 104 (2014), arXiv:1402.7088.
[54] M. A. Scheel, M. Boyle, T. Chu, L. E. Kidder, K. D. Matthews, and H. P. Pfeiffer, PRD **79**, 024003 (2009), arXiv:0810.1767 [gr-qc].
[55] S. Ossokine, M. Boyle, L. E. Kidder, H. P. Pfeiffer, M. A. Scheel, and B. Szilágyi, PRD **92**, 104028 (2015), arXiv:1502.01747 [gr-qc].
[56] J. Veitch *et al.*, PRD **91**, 042003 (2015), arXiv:1409.7215 [gr-qc].
[57] K. Chatziioannou, N. Cornish, A. Klein, and N. Yunes, (2016), in preparation.
[58] G. van Rossum and J. de Boer, CWI Quarterly **4**, 283 (1991).
[59] S. van der Walt, S. Colbert, and G. Varoquaux, Computing in Science Engineering **13**, 22 (2011).
[60] E. Jones, T. Oliphant, P. Peterson, *et al.*, *SciPy: Open source scientific tools for Python* (2001) www.scipy.org.
[61] J. D. Hunter, Computing in Science and Engineering **9**, 90 (2007).
[62] S. Oller-Moreno, *parmap: Easy parallelization in Python* (2015) pypi.python.org/pypi/parmap.
[63] D. Gerosa, M. Kesden, R. O'Shaughnessy, A. Klein, E. Berti, U. Sperhake, and D. Trifirò, PRL **115**, 141102 (2015), arXiv:1506.09116 [gr-qc].
[64] D. Gerosa, B. Veronesi, G. Lodato, and G. Rosotti, MNRAS **451**, 3941 (2015), arXiv:1503.06807.
[65] D. Trifirò, R. O'Shaughnessy, D. Gerosa, E. Berti, M. Kesden, T. Littenberg, and U. Sperhake, PRD **93**, 044071 (2016), arXiv:1507.05587 [gr-qc].
[66] D. Gerosa and C. J. Moore, (2016), in preparation.
[67] A. Gallant, *pdoc* (2014) pypi.python.org/pypi/pdoc.
[68] P. C. Peters, Physical Review **136**, 1224 (1964).
[69] B. P. Abbott *et al.*, (2016), arXiv:1602.03840 [gr-qc].
[70] A. Sesana, ApJ **719**, 851 (2010), arXiv:1006.0730 [astro-ph.CO].
[71] C. Roedig and A. Sesana, Journal of Physics Conference Series **363**, 012035 (2012), arXiv:1111.3742.
[72] E. Poisson, A. Pound, and I. Vega, Living Reviews in Relativity **14** (2011), arXiv:1102.0529 [gr-qc].
[73] É. Racine, PRD **78**, 044021 (2008), arXiv:0803.1820 [gr-qc].
[74] C. O. Lousto and Y. Zlochower, PRD **89**, 104052 (2014), arXiv:1312.5775 [gr-qc].

[75] J. Vosmera and D. Gerosa, (2016), in preparation.
[76] J. D. Schnittman, PRD **70**, 124020 (2004), astro-ph/0409174.
[77] A. Buonanno, Y. Chen, and T. Damour, PRD **74**, 104005 (2006), gr-qc/0508067.
[78] D. Gerosa, R. O'Shaughnessy, M. Kesden, E. Berti, and U. Sperhake, PRD **89**, 124025 (2014), arXiv:1403.7147 [gr-qc].
[79] T. Damour, PRD **64**, 124013 (2001), gr-qc/0103018.
[80] A. Gupta and A. Gopakumar, Classical and Quantum Gravity **31**, 105017 (2014), arXiv:1312.0217 [gr-qc].
[81] A. C. M. Correia, MNRAS **457**, L49 (2016), arXiv:1511.01890 [gr-qc].
[82] X. Zhao, M. Kesden, and D. Gerosa, (2016), in preparation.
[83] K. G. Arun, A. Buonanno, G. Faye, and E. Ochsner, PRD **79**, 104023 (2009), arXiv:0810.5336 [gr-qc].
[84] K. G. Arun, A. Buonanno, G. Faye, and E. Ochsner, PRD **84**, 049901 (2011).
[85] G. Faye, L. Blanchet, and A. Buonanno, PRD **74**, 104033 (2006), gr-qc/0605139.
[86] L. Blanchet, A. Buonanno, and G. Faye, PRD **74**, 104034 (2006), gr-qc/0605140.
[87] L. E. Kidder, C. M. Will, and A. G. Wiseman, PRD **47**, R4183 (1993), gr-qc/9211025.
[88] E. Poisson, PRD **57**, 5287 (1998), gr-qc/9709032.
[89] L. Á. Gergely, PRD **61**, 024035 (2000), gr-qc/9911082.
[90] L. Blanchet, G. Faye, B. R. Iyer, and B. Joguet, PRD **65**, 061501 (2002), gr-qc/0105099.
[91] L. Blanchet, G. Faye, B. R. Iyer, and B. Joguet, PRD **71**, 129902 (2005).
[92] L. Blanchet, T. Damour, G. Esposito-Farèse, and B. R. Iyer, PRL **93**, 091101 (2004), gr-qc/0406012.
[93] S. Marsat, A. Bohé, G. Faye, and L. Blanchet, Classical and Quantum Gravity **30**, 055007 (2013), arXiv:1210.4143 [gr-qc].
[94] A. Bohé, S. Marsat, G. Faye, and L. Blanchet, Classical and Quantum Gravity **30**, 075017 (2013), arXiv:1212.5520 [gr-qc].
[95] A. Bohé, G. Faye, S. Marsat, and E. K. Porter, Classical and Quantum Gravity **32**, 195010 (2015), arXiv:1501.01529 [gr-qc].
[96] T. Damour, P. Jaranowski, and G. Schäfer, PRD **89**, 064058 (2014), arXiv:1401.4548 [gr-qc].
[97] L. Bernard, L. Blanchet, A. Bohé, G. Faye, and S. Marsat, (2015), arXiv:1512.02876 [gr-qc].
[98] A. Hindmarsh, *ODEPACK, a systematized collection of ODE solvers* (Lawrence Livermore National Laboratory, 1982).
[99] V. Kalogera, ApJ **541**, 319 (2000), astro-ph/9911417.
[100] T. Bogdanović, C. S. Reynolds, and M. C. Miller, ApJ **661**, L147 (2007), astro-ph/0703054.
[101] M. Dotti, M. Volonteri, A. Perego, M. Colpi, M. Ruszkowski, and F. Haardt, MNRAS **402**, 682 (2010), arXiv:0910.5729 [astro-ph.HE].
[102] G. Lodato and D. Gerosa, MNRAS **429**, L30 (2013), arXiv:1211.0284.
[103] M. C. Miller and J. H. Krolik, ApJ **774**, 43 (2013), arXiv:1307.6569 [astro-ph.HE].
[104] G. Cowan, *Statistical data analysis* (Oxford: Clarendon Press, 1997).
[105] A. Buonanno, B. R. Iyer, E. Ochsner, Y. Pan, and B. S. Sathyaprakash, PRD **80**, 084043 (2009), arXiv:0907.0700 [gr-qc].
[106] M. Campanelli, C. O. Lousto, H. Nakano, and Y. Zlochower, PRD **79**, 084010 (2009), arXiv:0808.0713 [gr-qc].
[107] M. Kesden, PRD **78**, 084030 (2008), arXiv:0807.3043.
[108] E. Berti, V. Cardoso, J. A. Gonzalez, U. Sperhake, M. Hannam, S. Husa, and B. Brügmann, PRD **76**, 064034 (2007), gr-qc/0703053.
[109] W. Tichy and P. Marronetti, PRD **78**, 081501 (2008), arXiv:0807.2985 [gr-qc].
[110] E. Barausse, V. Morozova, and L. Rezzolla, ApJ **758**, 63 (2012), arXiv:1206.3803 [gr-qc].
[111] J. M. Bardeen, in *Black Holes (Les Astres Occlus)*, edited by C. Dewitt and B. S. Dewitt (1973) pp. 215–239.
[112] E. Barausse and L. Rezzolla, ApJ **704**, L40 (2009), arXiv:0904.2577 [gr-qc].
[113] A. Buonanno, L. E. Kidder, and L. Lehner, PRD **77**, 026004 (2008), arXiv:0709.3839.
[114] L. Rezzolla, E. Barausse, E. N. Dorband, D. Pollney, C. Reisswig, J. Seiler, and S. Husa, PRD **78**, 044002 (2008), arXiv:0712.3541 [gr-qc].
[115] D. Merritt, M. Milosavljević, M. Favata, S. A. Hughes, and D. E. Holz, ApJ **607**, L9 (2004), astro-ph/0402057.
[116] M. Campanelli, C. Lousto, Y. Zlochower, and D. Merritt, ApJ **659**, L5 (2007), gr-qc/0701164.
[117] J. A. González, U. Sperhake, B. Brügmann, M. Hannam, and S. Husa, PRL **98**, 091101 (2007), gr-qc/0610154.
[118] C. O. Lousto and Y. Zlochower, PRD **77**, 044028 (2008), arXiv:0708.4048 [gr-qc].
[119] C. O. Lousto, Y. Zlochower, M. Dotti, and M. Volonteri, PRD **85**, 084015 (2012), arXiv:1201.1923 [gr-qc].
[120] C. O. Lousto and Y. Zlochower, PRD **87**, 084027 (2013), arXiv:1211.7099 [gr-qc].
[121] J. A. González, M. Hannam, U. Sperhake, B. Brügmann, and S. Husa, PRL **98**, 231101 (2007), gr-qc/0702052.
[122] C. O. Lousto and Y. Zlochower, PRL **107**, 231102 (2011), arXiv:1108.2009 [gr-qc].
[123] B. Brügmann, J. A. González, M. Hannam, S. Husa, and U. Sperhake, PRD **77**, 124047 (2008), arXiv:0707.0135 [gr-qc].
[124] C. O. Lousto and Y. Zlochower, PRD **79**, 064018 (2009), arXiv:0805.0159 [gr-qc].
[125] E. Berti, M. Kesden, and U. Sperhake, PRD **85**, 124049 (2012), arXiv:1203.2920 [astro-ph.HE].
[126] M. Volonteri, K. Gültekin, and M. Dotti, MNRAS **404**, 2143 (2010), arXiv:1001.1743.
[127] J. Healy, C. O. Lousto, and Y. Zlochower, PRD **90**, 104004 (2014), arXiv:1406.7295 [gr-qc].
[128] Y. Zlochower and C. O. Lousto, PRD **92**, 024022 (2015), arXiv:1503.07536 [gr-qc].
[129] S. Husa, S. Khan, M. Hannam, M. Pürrer, F. Ohme, X. J. Forteza, and A. Bohé, PRD **93**, 044006 (2016), arXiv:1508.07250 [gr-qc].
[130] K. W. Smith, *Cython - A Guide for Python Programmers* (O'Reilly Media, 2015).
[131] J. Bovy, ApJS **216**, 29 (2015), arXiv:1412.3451.

```
print "\n *Parameter selection at finite separations*"
q=0.8   # Must be q<=1. Check documentation for q=1.
chi1=1. # Must be chi1<=1
chi2=1. # Must be chi2<=1
M,m1,m2,S1,S2=precession.get_fixed(q,chi1,chi2) # Total-mass units M=1
print "We study a binary with\n\tq=%.3f  m1=%.3f  m2=%.3f\n\tchi1=%.3f  S1=%.3f\n\tchi2=%.3f  S2=%.3f" %(q,m1,m2,chi1,S1,chi2,S2)
r=100*M # Must be r>10M for PN to be valid
print "at separation\n\tr=%.3f" %r
xi_min,xi_max=precession.xi_lim(q,S1,S2)
Jmin,Jmax=precession.J_lim(q,S1,S2,r)
Sso_min,Sso_max=precession.Sso_limits(S1,S2)
print "The geometrical limits on xi,J and S are\n\t%.3f<=xi<=%.3f\n\t%.3f<=J<=%.3f\n\t%.3f<=S<=%.3f"
 ↪  %(xi_min,xi_max,Jmin,Jmax,Sso_min,Sso_max)
J= (Jmin+Jmax)/2.
print "We select a value of J\n\tJ=%.3f " %J
St_min,St_max=precession.St_limits(J,q,S1,S2,r)
print "This constrains the range of S to\n\t%.3f<=S<=%.3f" %(St_min,St_max)
xi_low,xi_up=precession.xi_allowed(J,q,S1,S2,r)
print "The allowed range of xi is\n\t%.3f<=xi<=%.3f" %(xi_low,xi_up)
xi=(xi_low+xi_up)/2.
print "We select a value of xi\n\txi=%.3f" %xi
test=(J>=min(precession.J_allowed(xi,q,S1,S2,r)) and J<=max(precession.J_allowed(xi,q,S1,S2,r)))
print "Is our couple (xi,J) consistent?", test
Sb_min,Sb_max=precession.Sb_limits(xi,J,q,S1,S2,r)
print "S oscillates between\n\tS-=%.3f\n\tS+=%.3f" %(Sb_min,Sb_max)
S=(Sb_min+Sb_max)/2.
print "We select a value of S between S- and S+\n\tS=%.3f" %S
t1,t2,dp,t12=precession.parametric_angles(S,J,xi,q,S1,S2,r)
print "The angles describing the spin orientations are\n\t(theta1,theta2,DeltaPhi)=(%.3f,%.3f,%.3f)" %(t1,t2,dp)
xi,J,S = precession.from_the_angles(t1,t2,dp,q,S1,S2,r)
print "From the angles one can recovery\n\t(xi,J,S)=(%.3f,%.3f,%.3f)" %(xi,J,S)

print "\n *Features of spin precession*"
t1_dp0,t2_dp0,t1_dp180,t2_dp180=precession.resonant_finder(xi,q,S1,S2,r)
print "The spin-orbit resonances for these values of J and xi are\n\t(theta1,theta2)=(%.3f,%.3f) for
 ↪  DeltaPhi=0\n\t(theta1,theta2)=(%.3f,%.3f) for DeltaPhi=pi" %(t1_dp0,t2_dp0,t1_dp180,t2_dp180)
tau = precession.precession_period(xi,J,q,S1,S2,r)
print "We integrate dt/dS to calculate the precessional period\n\ttau=%.3f" %tau
alpha = precession.alphaz(xi,J,q,S1,S2,r)
print "We integrate Omega*dt/dS to find\n\talpha=%.3f" %alpha
morphology = precession.find_morphology(xi,J,q,S1,S2,r)
if morphology==-1: labelm="Librating about DeltaPhi=0"
elif morphology==1: labelm="Librating about DeltaPhi=pi"
elif morphology==0: labelm="Circulating"
print "The precessional morphology is: "+labelm
sys.stdout = os.devnull # Ignore warnings
phase,xi_transit_low,xi_transit_up=precession.phase_xi(J,q,S1,S2,r)
sys.stdout = sys.__stdout__ # Restore warnings
if phase==-1: labelp="a single DeltaPhi~pi phase"
elif phase==2: labelp="two DeltaPhi~pi phases, a Circulating phase"
elif phase==3: labelp="a DeltaPhi~0, a Circulating, a DeltaPhi~pi phase"
print "The coexisting phases are: "+labelp

print "\n *Parameter selection at infinitely large separation*"
print "We study a binary with\n\tq=%.3f   m1=%.3f  m2=%.3f\n\tchi1=%.3f  S1=%.3f\n\tchi2=%.3f  S2=%.3f" %(q,m1,m2,chi1,S1,chi2,S2)
print "at infinitely large separation"
kappainf_min,kappainf_max=precession.kappainf_lim(S1,S2)
print "The geometrical limits on xi and kappa_inf are\n\t%.3f<=xi<=%.3f\n\t %.3f<=kappa_inf<=%.3f"
 ↪  %(xi_min,xi_max,kappainf_min,kappainf_max)
print "We select a value of xi\n\txi=%.3f" %xi
kappainf_low,kappainf_up=precession.kappainf_allowed(xi,q,S1,S2)
print "This constrains the range of kappa_inf to\n\t%.3f<=kappa_inf<=%.3f" %(kappainf_low,kappainf_up)
kappainf=(kappainf_low+kappainf_up)/2.
print "We select a value of kappa_inf\n\tkappa_inf=%.3f" %kappainf
test=(xi>=min(precession.xiinf_allowed(kappainf,q,S1,S2)) and xi<=max(precession.xiinf_allowed(kappainf,q,S1,S2)))
print "Is our couple (xi,kappa_inf) consistent?", test
t1_inf,t2_inf=precession.thetas_inf(xi,kappainf,q,S1,S2)
print "The asymptotic (constant) values of theta1 and theta2 are\n\t(theta1_inf,theta2_inf)=(%.3f,%.3f)" %(t1_inf,t2_inf)
xi,kappainf = precession.from_the_angles_inf(t1_inf,t2_inf,q,S1,S2)
print "From the angles one can recovery\n\t(xi,kappa_inf)=(%.3f,%.3f)" %(xi,kappainf)
```

FIG. 2. Source code of `test.parameter_selection`, described in Sec. VI A. The screen output is reported in Fig. 8. In this example we (i) select consistent parameters at finite separation, (ii) compute several quantities to characterize the precessional dynamics and (iii) select consistent parameters at infinitely large separation. This test is run typing `precession.test.parameter_selection()`.

```
fig=pylab.figure(figsize=(6,6))      # Create figure object and axes
ax_t1=fig.add_axes([0,1.95,0.9,0.5]) # first (top)
ax_t2=fig.add_axes([0,1.3,0.9,0.5])  # second
ax_dp=fig.add_axes([0,0.65,0.9,0.5]) # third
ax_t12=fig.add_axes([0,0,0.9,0.5])   # fourth (bottom)

q=0.7    # Mass ratio. Must be q<=1.
chi1=0.6 # Primary spin. Must be chi1<=1
chi2=1.  # Secondary spin. Must be chi2<=1
M,m1,m2,S1,S2=precession.get_fixed(q,chi1,chi2) # Total-mass units M=1
r=20*M   # Separation. Must be r>10M for PN to be valid
J=0.94   # Magnitude of J: Jmin<J<Jmax as given by J_lim
xi_vals=[-0.41,-0.3,-0.22] # Effective spin: xi_low<xi<xi_up as given by xi_allowed

for xi,color in zip(xi_vals,['blue','green','red']): # Loop over three binaries

    tau = precession.precession_period(xi,J,q,S1,S2,r) # Period
    morphology = precession.find_morphology(xi,J,q,S1,S2,r) # Morphology
    if morphology==-1: labelm="${\\rm L}0$"
    elif morphology==1: labelm="${\\rm L}\\pi$"
    elif morphology==0: labelm="${\\rm C}$"
    Sb_min,Sb_max=precession.Sb_limits(xi,J,q,S1,S2,r) # Limits in S
    S_vals = numpy.linspace(Sb_min,Sb_max,1000) # Create array, from S- to S+
    S_go=S_vals # First half of the precession cycle: from S- to S+
    t_go=map(lambda x: precession.t_of_S(S_go[0],x, Sb_min,Sb_max,xi,J,q,S1,S2,r,0,sign=-1.),S_go) # Compute time values. Assume t=0
     ↪  at S-
    t1_go,t2_go,dp_go,t12_go=zip(*[precession.parametric_angles(S,J,xi,q,S1,S2,r) for S in S_go]) # Compute the angles.
    dp_go=[-dp for dp in dp_go] # DeltaPhi<=0 in the first half of the cycle
    S_back=S_vals[::-1] # Second half of the precession cycle: from S+ to S-
    t_back=map(lambda x: precession.t_of_S(S_back[0],x, Sb_min,Sb_max, xi,J,q,S1,S2,r,t_go[-1],sign=1.),S_back) # Compute time, start
     ↪  from the last point of the first half t_go[-1]
    t1_back,t2_back,dp_back,t12_back=zip(*[precession.parametric_angles(S,J,xi,q,S1,S2,r) for S in S_back]) # Compute the angles.
     ↪  DeltaPhi>=0 in the second half of the cycle

    for ax,vec_go,vec_back in zip([ax_t1,ax_t2,ax_dp,ax_t12], [t1_go,t2_go,dp_go,t12_go], [t1_back,t2_back,dp_back,t12_back]): # Plot
     ↪  all curves
        ax.plot([t/tau for t in t_go],vec_go,c=color,lw=2,label=labelm)
        ax.plot([t/tau for t in t_back],vec_back,c=color,lw=2)

# Options for nice plotting
(...)
fig.savefig("spin_angles.pdf",bbox_inches='tight') # Save pdf file
```

FIG. 3. Source code of `test.spin_angles`, described in Sec. VI B. The resulting plot is shown in Fig. 9. This example illustrates how to study the evolution of the angles θ_1, θ_2, $\Delta\Phi$ and θ_{12} over a single precession cycle $S_- \to S_+ \to S_-$. The precessional dynamics is first parametrized using S, and then plotted in terms of the time t integrating dS/dt from Eq. (17). We assume $S = S-$ at $t = 0$ and match the two halves of the precession cycle at $S = S+$. Note that the sign of $\Delta\Phi$ has to be specified by the user. Three binaries are considered here; their precessional morphology is evaluated and used to fill the plot legend. This test is run typing `precession.test.spin_angles()`. Additional plotting options present in the source code have been omitted.

```
fig=pylab.figure(figsize=(6,6))      #Create figure object and axes
ax_tS=fig.add_axes([0,0,0.6,0.6])    #bottom-left
ax_td=fig.add_axes([0.65,0,0.3,0.6]) #bottom-right
ax_Sd=fig.add_axes([0,0.65,0.6,0.3]) #top-left

q=0.5    # Mass ratio. Must be q<=1.
chi1=0.3 # Primary spin. Must be chi1<=1
chi2=0.9 # Secondary spin. Must be chi2<=1
M,m1,m2,S1,S2=precession.get_fixed(q,chi1,chi2) # Total-mass units M=1
r=200.*M # Separation. Must be r>10M for PN to be valid
J=3.14   # Magnitude of J: Jmin<J<Jmax as given by J_lim
xi=-0.01 # Effective spin: xi_low<xi<xi_up as given by xi_allowed
Sb_min,Sb_max=precession.Sb_limits(xi,J,q,S1,S2,r) # Limits in S
tau=precession.precession_period(xi,J,q,S1,S2,r)   # Precessional period
d=2000   # Size of the statistical sample

precession.make_temp() # Create store directory, if necessary
filename=precession.storedir+"/phase_resampling.dat" # Output file name
if not os.path.isfile(filename): # Compute and store data if not present
    out=open(filename,"w")
    out.write("# q chi1 chi2 r J xi d\n") # Write header
    out.write( "# "+' '.join([str(x) for x in (q,chi1,chi2,r,J,xi,d)])+"\n")

    # S and t values for the S(t) plot
    S_vals=numpy.linspace(Sb_min,Sb_max,d)
    t_vals=numpy.array([abs(precession.t_of_S(Sb_min,S,Sb_min,Sb_max,xi,J,q,S1,S2,r)) for S in S_vals])
    # Sample values of S from |dt/dS|. Distribution should be flat in t.
    S_sample=numpy.array([precession.samplingS(xi,J,q,S1,S2,r) for i in range(d)])
    t_sample=numpy.array([abs(precession.t_of_S(Sb_min,S,Sb_min,Sb_max,xi,J,q,S1,S2,r)) for S in S_sample])
    # Continuous distributions (normalized)
    S_distr=numpy.array([2.*abs(precession.dtdS(S,xi,J,q,S1,S2,r)/tau) for S in S_vals])
    t_distr=numpy.array([2./tau for t in t_vals])

    out.write("# S_vals t_vals S_sample t_sample S_distr t_distr\n")
    for Sv,tv,Ss,ts,Sd,td in zip(S_vals,t_vals,S_sample,t_sample,S_distr,t_distr):
        out.write(' '.join([str(x) for x in (Sv,tv,Ss,ts,Sd,td)])+"\n")
    out.close()
else:  # Read
    S_vals,t_vals,S_sample,t_sample,S_distr,t_distr=numpy.loadtxt(filename,unpack=True)

# Rescale all time values by 10^-6, for nicer plotting
tau*=1e-6; t_vals*=1e-6; t_sample*=1e-6; t_distr/=1e-6

ax_tS.plot(S_vals,t_vals,c='blue',lw=2)  # S(t) curve
ax_td.plot(t_distr,t_vals,lw=2.,c='red') # Continous distribution P(t)
ax_Sd.plot(S_vals,S_distr,lw=2.,c='red') # Continous distribution P(S)
ax_td.hist(t_sample,bins=60,range=(0,tau/2.),normed=True,histtype='stepfilled', color="blue",alpha=0.4,orientation="horizontal") #
 ↪  Histogram P(t)
ax_Sd.hist(S_sample,bins=60,range=(Sb_min,Sb_max),normed=True,histtype='stepfilled', color="blue",alpha=0.4) # Histogram P(S)

# Options for nice plotting
(...)
fig.savefig("phase_resampling.pdf",bbox_inches='tight') # Save pdf file
```

FIG. 4. Source code of `test.phase_resampling`, described in Sec. VI C. The resulting plot is shown in Fig. 10. We extract N=2000 values of the precessional phase S from the probability distribution $P(S) = 2|dS/dt|^{-1}/\tau$ in $[S-, S+]$. The procedure is illustrated in Sec. IV C and is a key step to perform precession-averaged inspirals. We verify that the distribution $t(S)$ constructed from the sampled values of S is uniform in $[0, \tau/2]$. This test is run typing `precession.test.phase_resampling()`. Data are stored in `precession.storedir`. Additional plotting options present in the source code have been omitted.

```
q=0.9      # Mass ratio. Must be q<=1.
chi1=0.5   # Primary spin. Must be chi1<=1
chi2=0.5   # Secondary spin. Must be chi2<=1
print "We study a binary with\n\tq=%.3f, chi1=%.3f, chi2=%.3f" %(q,chi1,chi2)
M,m1,m2,S1,S2=precession.get_fixed(q,chi1,chi2) # Total-mass units M=1
ri=1000*M  # Initial separation.
rf=10.*M   # Final separation.
rt=100.*M  # Intermediate separation for hybrid evolution.
r_vals=numpy.logspace(numpy.log10(ri),numpy.log10(rf),10) # Output requested
t1i=numpy.pi/4.; t2i=numpy.pi/4.; dpi=numpy.pi/4. # Initial configuration
xii,Ji,Si=precession.from_the_angles(t1i,t2i,dpi,q,S1,S2,ri)
print "Configuration at ri=%.0f\n\t(xi,J,S)=(%.3f,%.3f,%.3f)\n\t(theta1,theta2,deltaphi)=(%.3f,%.3f,%.3f)"
 ↪  %(ri,xii,Ji,Si,t1i,t2i,dpi)

print " *Orbit-averaged evolution*"
print "Evolution ri=%.0f --> rf=%.0f" %(ri,rf)
Jf,xif,Sf=precession.orbit_averaged(Ji,xii,Si,r_vals,q,S1,S2)
print "\t(xi,J,S)=(%.3f,%.3f,%.3f)" %(xif[-1],Jf[-1],Sf[-1])
t1f,t2f,dpf=precession.orbit_angles(t1i,t2i,dpi,r_vals,q,S1,S2)
print "\t(theta1,theta2,deltaphi)=(%.3f,%.3f,%.3f)" %(t1f[-1],t2f[-1],dpf[-1])
Jvec,Lvec,S1vec,S2vec,Svec=precession.Jframe_projection(xii,Si,Ji,q,S1,S2,ri)
Lxi,Lyi,Lzi=Lvec; S1xi,S1yi,S1zi=S1vec; S2xi,S2yi,S2zi=S2vec
Lx,Ly,Lz,S1x,S1y,S1z,S2x,S2y,S2z=precession.orbit_vectors(Lxi,Lyi,Lzi,S1xi,S1yi,S1zi,S2xi,S2yi,S2zi,r_vals,q)
print "\t(Lx,Ly,Lz)=(%.3f,%.3f,%.3f)\n\t(S1x,S1y,S1z)=(%.3f,%.3f,%.3f)\n\t(S2x,S2y,S2z)=(%.3f,%.3f,%.3f)"
 ↪  %(Lx[-1],Ly[-1],Lz[-1],S1x[-1],S1y[-1],S1z[-1],S2x[-1],S2y[-1],S2z[-1])

print " *Precession-averaged evolution*"
print "Evolution ri=%.0f --> rf=%.0f" %(ri,rf)
Jf=precession.evolve_J(xii,Ji,r_vals,q,S1,S2)
print "\t(xi,J,S)=(%.3f,%.3f,-)" %(xii,Jf[-1])
t1f,t2f,dpf=precession.evolve_angles(t1i,t2i,dpi,r_vals,q,S1,S2)
print "\t(theta1,theta2,deltaphi)=(%.3f,%.3f,%.3f)" %(t1f[-1],t2f[-1],dpf[-1])
print "Evolution ri=%.0f --> infinity" %ri
kappainf=precession.evolve_J_backwards(xii,Jf[-1],rf,q,S1,S2)
print "\tkappainf=%.3f" %kappainf
Jf=precession.evolve_J_infinity(xii,kappainf,r_vals,q,S1,S2)
print "Evolution infinity --> rf=%.0f" %rf
print "\tJ=%.3f" %Jf[-1]

print " *Hybrid evolution*"
print "Prec.Av. infinity --> rt=%.0f & Orb.Av. rt=%.0f --> rf=%.0f" %(rt,rt,rf)
t1f,t2f,dpf=precession.hybrid(xii,kappainf,r_vals,q,S1,S2,rt)
print "\t(theta1,theta2,deltaphi)=(%.3f,%.3f,%.3f)" %(t1f[-1],t2f[-1],dpf[-1])

print " *Properties of the BH remnant*"
Mfin=precession.finalmass(t1f[-1],t2f[-1],dpf[-1],q,S1,S2)
print "\tM_f=%.3f" %Mfin
chifin=precession.finalspin(t1f[-1],t2f[-1],dpf[-1],q,S1,S2)
print "\tchi_f=%.3f, S_f=%.3f" %(chifin,chifin*Mfin**2)
vkick=precession.finalkick(t1f[-1],t2f[-1],dpf[-1],q,S1,S2)
print "\tvkick=%.5f" %(vkick) # Geometrical units c=1
```

FIG. 5. Source code of `test.PNwrappers`, described in Sec. VI D. The screen output is reported in Fig. 11. This example shows how to use the various routines to perform PN inspiral. After specifying a BH binary at r_i, we evolve it down to r_f using both orbit-averaged and precession-averaged integrations. We then extract the asymptotic configuration κ_∞ and show how to match precession-averaged and orbit-averaged evolutions to construct hybrid inspirals. We also estimate mass, spin and recoil of the post-merger BH. This test is run typing `precession.test.PNwrappers()`. Data are stored in the directory specified through `precession.storedir`.

```
fig=pylab.figure(figsize=(6,6)) # Create figure object and axes
L,Ws,Wm,G=0.85,0.15,0.3,0.03    # Sizes
ax_Sd=fig.add_axes([0,0,L,Ws])              # bottom-small
ax_S=fig.add_axes([0,Ws,L,Wm])              # bottom-main
ax_Jd=fig.add_axes([0,Ws+Wm+G,L,Ws])        # middle-small
ax_J=fig.add_axes([0,Ws+Ws+Wm+G,L,Wm])      # middle-main
ax_xid=fig.add_axes([0,2*(Ws+Wm+G),L,Ws])   # top-small
ax_xi=fig.add_axes([0,Ws+2*(Ws+Wm+G),L,Wm]) # top-main

q=0.8      # Mass ratio. Must be q<=1.
chi1=0.6   # Primary spin. Must be chi1<=1
chi2=1.    # Secondary spin. Must be chi2<=1
M,m1,m2,S1,S2=precession.get_fixed(q,chi1,chi2) # Total-mass units M=1
ri=100.*M  # Initial separation.
rf=10.*M   # Final separation.
r_vals=numpy.linspace(ri,rf,1001) # Output requested
Ji=2.24    # Magnitude of J: Jmin<J<Jmax as given by J_lim
xi=-0.5    # Effective spin: xi_low<xi<xi_up as given by xi_allowed

Jf_P=precession.evolve_J(xi,Ji,r_vals,q,S1,S2) # Pr.av. integration
Sf_P=[precession.samplingS(xi,J,q,S1,S2,r) for J,r in zip(Jf_P[0::10],r_vals[0::10])] # Resample S (reduce output for clarity)
Sb_min,Sb_max= zip(*[precession.Sb_limits(xi,J,q,S1,S2,r) for J,r in zip(Jf_P,r_vals)]) # Envelopes
S=numpy.average([precession.Sb_limits(xi,Ji,q,S1,S2,ri)]) # Initialize S
Jf_O,xif_O,Sf_O=precession.orbit_averaged(Ji,xi,S,r_vals,q,S1,S2) # Orb.av. integration

Pcol,Ocol,Dcol='blue','red','green'
Pst,Ost='solid','dashed'
ax_xi.axhline(xi,c=Pcol,ls=Pst,lw=2)         # Plot xi, pr.av. (constant)
ax_xi.plot(r_vals,xif_O,c=Ocol,ls=Ost,lw=2)  # Plot xi, orbit averaged
ax_xid.plot(r_vals,(xi-xif_O)/xi*1e11,c=Dcol,lw=2) # Plot xi deviations (rescaled)
ax_J.plot(r_vals,Jf_P,c=Pcol,ls=Pst,lw=2)    # Plot J, pr.av.
ax_J.plot(r_vals,Jf_O,c=Ocol,ls=Ost,lw=2)    # Plot J, orb.av
ax_Jd.plot(r_vals,(Jf_P-Jf_O)/Jf_O*1e3,c=Dcol,lw=2) # Plot J deviations (rescaled)
ax_S.scatter(r_vals[0::10],Sf_P,facecolor='none',edgecolor=Pcol) # Plot S, pr.av. (resampled)
ax_S.plot(r_vals,Sb_min,c=Pcol,ls=Pst,lw=2)  # Plot S, pr.av. (envelopes)
ax_S.plot(r_vals,Sb_max,c=Pcol,ls=Pst,lw=2)  # Plot S, pr.av. (envelopes)
ax_S.plot(r_vals,Sf_O,c=Ocol,ls=Ost,lw=2)    # Plot S, orb.av (evolved)
ax_Sd.plot(r_vals[0::10],(Sf_P-Sf_O[0::10])/Sf_O[0::10],c=Dcol,lw=2) # Plot S deviations

# Options for nice plotting
(...)
fig.savefig("compare_evolutions.pdf",bbox_inches='tight') # Save pdf file
```

FIG. 6. Source code of `test.compare_evolutions`, described in Sec. VI E. The resulting plot is shown in Fig. 12. We compare precession-averaged and orbit-averaged integrations of a single BH binary. We perform the two integrations from $r_i = 100M$ to $r_f = 10M$ and extract values of ξ, J and S along the inspiral. Relative differences between the two approaches are computed and plotted as a function of the binary separation. This test is run typing `precession.test.PNwrappers()`. Data are stored in the directory specified through `precession.storedir`. Additional plotting options present in the source code have been omitted.

```
BHsample=[] #  Construct a sample of BH binaries
N=100
for i in range(N):
    q=random.uniform(0,1)
    chi1=random.uniform(0,1)
    chi2=random.uniform(0,1)
    M,m1,m2,S1,S2=precession.get_fixed(q,chi1,chi2)
    t1=random.uniform(0,numpy.pi)
    t2=random.uniform(0,numpy.pi)
    dp=random.uniform(0,2.*numpy.pi)
    BHsample.append([q,S1,S2,t1,t2,dp])
q_vals,S1_vals,S2_vals,t1i_vals,t2i_vals,dpi_vals=zip(*BHsample) # Traspose python list

ri=1000*M      # Initial separation
rf=10*M        # Final separation
r_vals=[ri,rf] # Intermediate output separations not needed here

print "Integrating a sample of N=%.0f BH binaries from ri=%.0f to rf=%.0f using %.0f CPUs" %(N,ri,rf,multiprocessing.cpu_count()) #
 ↪  Parallel computation used by default
t0=time.time()
precession.orbit_angles(t1i_vals,t2i_vals,dpi_vals,r_vals,q_vals,S1_vals,S2_vals)
t=time.time()-t0
print "Orbit-averaged: parallel integrations\n\t total time t=%.3fs\n\t time per binary t/N=%.3fs" %(t,t/N)
t0=time.time()
precession.evolve_angles(t1i_vals,t2i_vals,dpi_vals,r_vals,q_vals,S1_vals,S2_vals)
t=time.time()-t0
print "Precession-averaged: parallel integrations\n\t total time t=%.3fs\n\t time per binary t/N=%.3fs" %(t,t/N)

precession.empty_temp() # Remove previous checkpoints
precession.CPUs=1       # Force serial computation
print "\nIntegrating a sample of N=%.0f BH binaries from ri=%.0f to rf=%.0f using %.0f CPU" %(len(BHsample),ri,rf,precession.CPUs)
t0=time.time()
precession.orbit_angles(t1i_vals,t2i_vals,dpi_vals,r_vals,q_vals,S1_vals,S2_vals)
t=time.time()-t0
print "Orbit-averaged: serial integrations\n\t total time t=%.3fs\n\t time per binary t/N=%.3fs" %(t,t/N)
t0=time.time()
precession.evolve_angles(t1i_vals,t2i_vals,dpi_vals,r_vals,q_vals,S1_vals,S2_vals)
t=time.time()-t0
print "Precession-averaged: serial integrations\n\t total time t=%.3fs\n\t time per binary t/N=%.3fs" %(t,t/N)
precession.empty_temp() # Remove previous checkpoints
```

FIG. 7. Source code of `test.timing` described in Sec. VI F; the screen output is reported in Fig. 13. We compute the CPU time needed to evolve a sample of $N = 100$ binaries from $r_i = 10^4 M$ to $r_f = 10M$ using both orbit-averaged and precession-averaged integrations. By default, PRECESSION performs PN inspirals in parallel using all available cores. The two computations are repeated enforcing a strictly serial execution. This test is run typing `precession.test.timing()`. Data are stored in the directory specified through `precession.storedir`.

```
 *Parameter selection at finite separations*
We study a binary with
        q=0.800     m1=0.556  m2=0.444
        chi1=1.000  S1=0.309
        chi2=1.000  S2=0.198
at separation
        r=100.000
The geometrical limits on xi,J and S are
        -1.000<=xi<=1.000
        1.963<=J<=2.975
        0.111<=S<=0.506
We select a value of J
        J=2.469
This constrains the range of S to
        0.111<=S<=0.506
The allowed range of xi is
        -0.131<=xi<=0.073
We select a value of xi
        xi=-0.029
Is our couple (xi,J) consistent? True
S oscillates between
        S-=0.114
        S+=0.448
We select a value of S between S- and S+
        S=0.281
The angles describing the spin orientations are
        (theta1,theta2,DeltaPhi)=(1.622,1.571,2.041)
From the angles one can recovery
        (xi,J,S)=(-0.029,2.469,0.281)

 *Features of spin precession*
The spin-orbit resonances for these values of J and xi are
        (theta1,theta2)=(1.100,2.254) for DeltaPhi=0
        (theta1,theta2)=(2.572,0.155) for DeltaPhi=pi
We integrate dt/dS to calculate the precessional period
        tau=3037166.882
We integrate Omega*dt/dS to find
        alpha=23.660
The precessional morphology is: Circulating
The coexisting phases are: a DeltaPhi~0, a Circulating, a
↪  DeltaPhi~pi phase

 *Parameter selection at infinitely large separation*
We study a binary with
        q=0.800     m1=0.556  m2=0.444
        chi1=1.000  S1=0.309
        chi2=1.000  S2=0.198
at infinitely large separation
The geometrical limits on xi and kappa_inf are
        -1.000<=xi<=1.000
         -0.506<=kappa_inf<=0.506
We select a value of xi
        xi=-0.029
This constrains the range of kappa_inf to
        -0.065<=kappa_inf<=0.033
We select a value of kappa_inf
        kappa_inf=-0.016
Is our couple (xi,kappa_inf) consistent? True
The asymptotic (constant) values of theta1 and theta2 are
        (theta1_inf,theta2_inf)=(1.623,1.571)
From the angles one can go back to
        (xi,kappa_inf)=(-0.029,-0.016)
```

FIG. 8. Screen output of `test.parameter_selection`, described in Sec. VI A. The source code is reported in Fig. 2. In this example we (i) select consistent parameters at finite separation, (ii) compute several quantities to characterize the precessional dynamics and (iii) select consistent parameters at infinitely large separation. Outputs have been rounded to 3 decimal digits for clarity. This test is run typing `precession.test.parameter_selection()`.

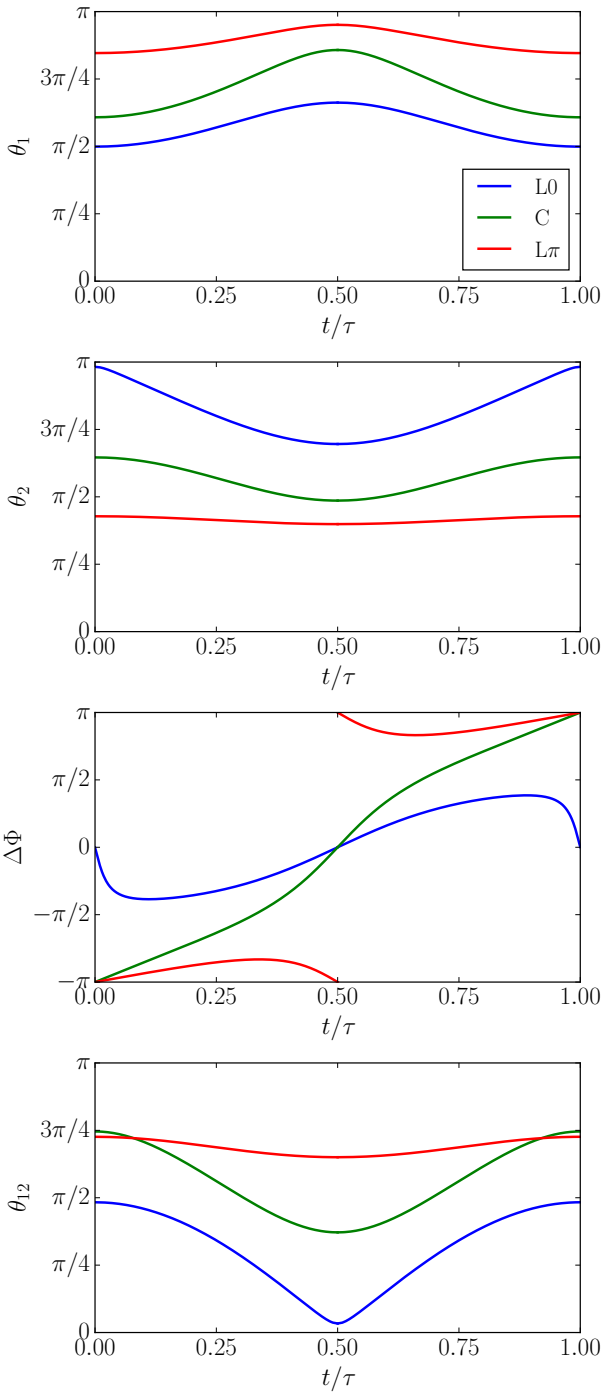

FIG. 9. Resulting plot obtained from `test.spin_angles`, described in Sec. VI B. The source code is reported in Fig. 3. We study the precessional dynamics of three binary BHs with mass ratio $q = 0.7$, dimensionless spin $\chi_1 = 0.6$, $\chi_2 = 1$, total angular momentum $J = 0.94M^2$ at separation $r = 20M$. The evolution of the angles θ_1, θ_2, $\Delta\Phi$ and θ_{12} (top to bottom) is plotted against the time t normalized to the precessional period τ. The configurations shown here are characterized by different values of the effective spin ξ and belong to the three different morphologies: the binary with $\xi = -0.41$ (blue) is librating about $\Delta\Phi = 0$ (L0); the binary with $\xi = -0.3$ (green) is circulating through the full range $\Delta\Phi \in [-\pi, \pi]$ (C), and the binary with $\xi = -0.22$ (red) is librating about $\Delta\Phi = \pm\pi$ (Lπ). This test is run typing `precession.test.spin_angles()`.

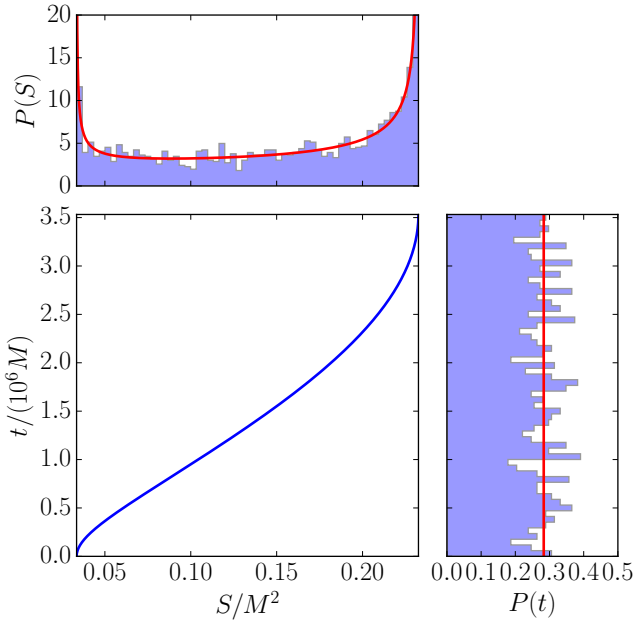

FIG. 10. Resulting plot obtained from `test.phase_resampling`, described in Sec. VI C. The source code is reported in Fig. 4. The bottom left panel shows the evolution of S on the precession time for a BH binary with $q = 0.5$, $\chi_1 = 0.3$, $\chi_2 = 0.9$, $J = 3.14M^2$, $\xi = -0.01$ and $r = 200M$. The binary evolves from $S_- \simeq 0.033$ ($t = 0$) to $S_+ \simeq 0.232$ ($t = \tau/2 \simeq 3.53 \times 10^6 M$). We extract a sample of $N = 2000$ values of S from a probability distribution proportional to $|dS/dt|^{-1}$. Histograms of the extracted distribution of S and t are shown in the top and right panels, respectively, where red lines mark the continuum limit. This procedure efficiently extracts BH binaries according to their time spent at each spin configuration and demonstrates the correct handling of the (integrable) singularities of $|dS/dt|^{-1}$ at $S_\pm$. This test is run typing `precession.test.phase_resampling()`.

```
We study a binary with q=0.900, chi1=0.500, chi2=0.500
Configuration at ri=1000
        (xi,J,S)=(0.354,8.063,0.241)
        (theta1,theta2,deltaphi)=(0.785,0.785,0.785)
 *Orbit-averaged evolution*
Evolution ri=1000 --> rf=10
        (xi,J,S)=(0.354,0.974,0.229)
        (theta1,theta2,deltaphi)=(1.032,0.397,-0.952)
        (Lx,Ly,Lz)=(0.077,-0.091,0.779)
        (S1x,S1y,S1z)=(-0.085,0.065,0.088)
        (S2x,S2y,S2z)=(0.013,0.031,0.107)
 *Precession-averaged evolution*
Evolution ri=1000 --> rf=10
        (xi,J,S)=(0.354,0.974,-)
        (theta1,theta2,deltaphi)=(0.573,0.978,-1.624)
Evolution ri=1000 --> infinity
        kappainf=0.178
Evolution infinity --> rf=10
        J=0.974
 *Hybrid evolution*
Prec.Av. infinity --> rt=100 & Orb.Av. rt=100 --> rf=10
        (theta1,theta2,deltaphi)=(0.639,0.926,-1.691)
 *Properties of the BH remnant*
        M_f=0.938
        chi_f=0.797, S_f=0.701
        vkick=0.00330
```

FIG. 11. Screen output of `test.PNwrappers`, described in Sec. VI D. The source code is reported in Fig. 5. After selecting a binary at the initial separation r_i, we (i) perform orbit-averaged integrations from r_i to a final separation r_f; (ii) perform precession-averaged integrations from r_i to r_f, from r_i to $r/M = \infty$ and from $r/M = \infty$ to r_f, (iii) perform hybrid integrations from $r/M = \infty$ to r_f matched at a separation threshold r_t and (iv) extract the properties of the BH remnant applying fitting formulae at r_f. Outputs have been rounded to 3 decimal digits for clarity; output lines regarding the location of the stored data files have been omitted. This test is run typing `precession.test.PNwrappers()`.

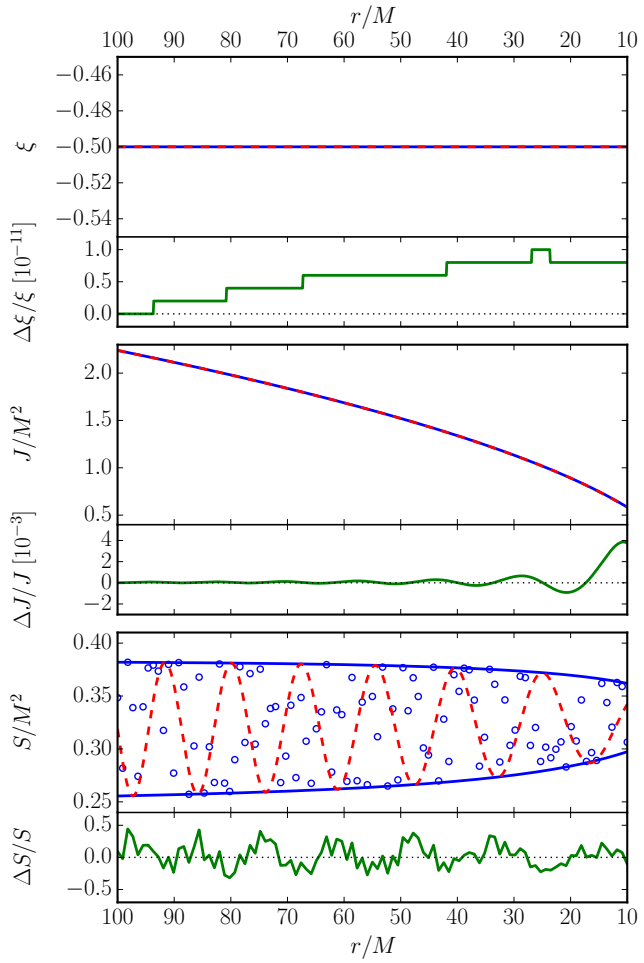

FIG. 12. Resulting plot obtained with the test function `test.compare_evolutions`, described in Sec. VI E. The source code is reported in Fig. 6. We choose a BH binary with $q = 0.8$, $\chi_1 = 0.6$, $\chi_2 = 1$, $J = 2.24M^2$ and $\xi = -0.5$ at $r_i = 100M$ and we compare its PN inspiral till $r_f = 10M$ using precession-averaged and orbit-averaged integrations. The evolutions of ξ (top), J (middle) and S (bottom) are shown in the larger subpanels. Results for J and ξ show excellent agreement between precession-averaged (solid blue) and orbit-averaged (dashed red) integrations. Precession-averaged integrations do not track the evolution of the total spin magnitude S, but estimates (blue circles) can be obtained by sampling S between S_- and S_+ (blue solid lines); results are in statistical agreement with the orbit-averaged result (dashed red line). Smaller subpanels (solid green lines) show the relative difference between the two approaches. This test is run typing `precession.test.compare_evolutions()`.

```
 *Integrating a sample of N=100 BH binaries from ri=10000 to
  ↪  rf=10 using 4 CPUs*
Orbit-averaged: parallel integrations
         total time t=5298.677s
         time per binary t/N=52.987s
Precession-averaged: parallel integrations
         total time t=73.742s
         time per binary t/N=0.737s
 *Integrating a sample of N=100 BH binaries from ri=10000 to
  ↪  rf=10 using 1 CPU*
Orbit-averaged: serial integrations
         total time t=18276.063s
         time per binary t/N=182.761s
Precession-averaged: serial integrations
         total time t=244.989s
         time per binary t/N=2.450s
```

FIG. 13. Screen output of `test.timing`, described in Sec. VI F. The source code is reported in Fig. 7. We time the performances of `orbit_angles` and `evolve_angles` using both parallel (first iteration) and serial (second iteration) computation. Times reported here are obtained using a 2013 Intel i5-3470 3.20GHz 4 cores CPU. Output lines regarding the location of the stored data files are omitted for clarity. This test is run typing `precession.test.timing()`.