Supporting Information: PyBERTHART: A relativistic real-time four-component TDDFT implementation using prototyping techniques based on Python.

Matteo De Santis,[†,‡] Loriano Storchi,[*,¶,‡] Leonardo Belpassi,[*,‡] Harry M. Quiney,[§] and Francesco Tarantelli[†,‡]

†*Dipartimento di Chimica, Biologia e Biotecnologie, Università degli Studi di Perugia, Via Elce di Sotto 8, 06123 Perugia, Italy*

‡*Istituto di Scienze e Tecnologie Chimiche (SCITEC), Consiglio Nazionale delle Ricerche c/o Dipartimento di Chimica, Biologia e Biotecnologie, Università degli Studi di Perugia, Via Elce di Sotto 8, 06123 Perugia, Italy*

¶*Dipartimento di Farmacia, Università degli Studi 'G. D'Annunzio', Via dei Vestini 31, 66100 Chieti, Italy*

§ *ARC Centre of Excellence for Advanced Molecular Imaging, School of Physics, The University of Melbourne, 3010 Victoria, Australia*

E-mail: loriano@storchi.org; leonardo.belpassi@cnr.it

Contents

1 PyBERTHA class

In the following we are reporting a concise description of all the pybertha class methods:

```
class pybertha
    Methods defined here:

    __init__(self, sopath='./bertha_wrapper.so')
        param: sopath is needed to specify the
        bertha_wrapper Shared Object file.

    density_to_cube(self, dens, fname,
              margin=10.0, drx=0.2, dry=0.2, drz=0.2)
        density_to_cube generates a cube file (named: fname)
        with the specified density (dens).

    finalize(self)
        To finalize and free all the allocated memory.

    get_densitydiff(self)
        The *get_densitydiff* returns the **densitydiff flag** value.

    get_dumpfiles(self)
        *get_dumpfiles* returns the dumpfiles flag value.

    get_erep(self)
        It returns nuclear repulsion energy.
```

```
get_etotal(self)
    get_etotal returns etotal so that etotal-(sfact*nocc)
    is equal to the total electronic energy.

get_fittcoefffname(self)
    get_fittcoefffname returns the density fitting
    coefficients filename.

get_fittfname(self)
    get_fittcoefffname returns the density fitting
    basis set filename.

get_fnameinput(self)
    get_fnameinput returns the BERTHA input filename.

get_fockctime(self)
    Returns the CPU time to perfom get_realtime_fock,
    not including the time needed to copy the arrays.

get_focktime(self)
    Returns the wall time to perfom get_realtime_fock,
    not including the time needed to copy the arrays.

get_mainrunctime(self)
    Returns the CPU time to perfom all the SCF iterations,
    not including the time needed to copy the arrays.
```

```
get_mainruntime(self)
    Returns the wall time to perfom all the SCF iterations,
    not including he time needed to copy the arrays.

get_ndim(self)
    get_ndim returns the matrix dimension.

get_nocc(self)
    get_nocc returns the number of occupied spinors.

get_nopen(self)
    *get_nopen* return number of unoccupied spinors

get_nshift(self)
    get_nshift returns the nshift value.

get_ovapfilename(self)
    get_ovapfilename returns the overlap matrix filename

get_realtime_dipolematrix(self, direction, normalise)
    get_realtime_dipolematrix computes the dipolematrix,
    the direction can be:
        - 2 = z direction
        - 3 = y direction
        - 4 = x direction

get_realtime_fock(self, densm)
```

```
    get_realtime_fock returns the Fock matrix given the
    density matrix densm.

get_sfact(self)
    *get_sfact* returns the level shift parameter.

get_thres(self)
    The *get_thres* returns the SCF convergence threshold.

get_vctfilename(self)
    *get_vctfilename* return the eigenvectors filename

get_verbosity(self)
    get_verbosity return the verbosity level.

init(self)
    Initialize method, it initializes all the variables
    and basix data. The user need to call this method
    before he/she can peform the main run.

realtime_init(self)
    realtime_init initializes all the data needed by the
    get_realtime_fock method.

run(self)
    This is the method to perform the SCF computation.
```

```
set_densitydiff(self, ini)
    Set densitydiff flag value.
    - If set to 1 the code will compute the maximum
      difference (elementwise) of the density matrix
      to the iteration n with respect to the one
      related to the previous iteration (n-1).

set_dumpfiles(self, ini)
    To specify the dumpfiles flag value:
        - if flag is 1

set_fittcoefffname(self, ini)
    To specify the filename where the density fitting
    coefficients will be written if the dumpfiles flag
    is equal to 1.

set_fittfname(self, ini)
    To specify the density fitting basis set filename.

set_fnameinput(self, ini)
    To specify the BERTHA input filename.

set_ovapfilename(self, ini)
    To specify the filename where the overlap matrix will
    be written if **dumpfiles flag** is equal to 1.

set_thres(self, ini)
```

Set the SCF convergence threshold.

set_vctfilename(self, ini)

To specify the filename where the eigenvectors will be written if **dumpfiles flag** is equal to 1.

set_verbosity(self, ini)

To set the verbosty level:

- 0 the program will run in a silent modes
- -1 only basic information are printed-out
- 1 all the details about the ongoing simlutaion are printed-out

2 Comparison bewteen standard Fourier Transform and using Padé approximant.

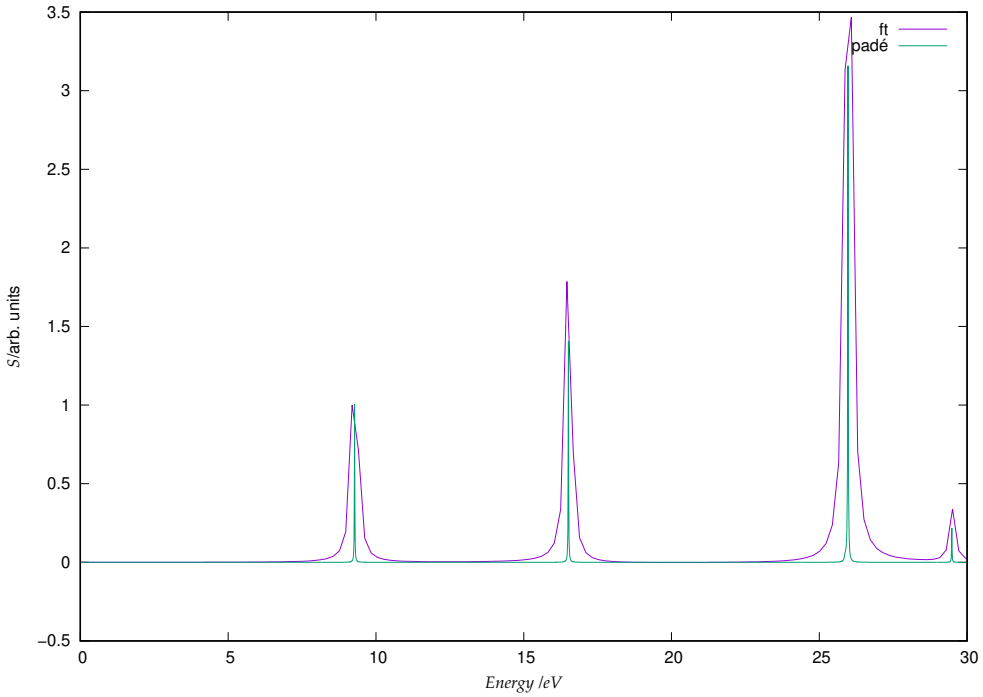

Figure S1: RT-TDDFT absorpition spectra obtained perturbing the system with a z-polarized kick. Green line: 2001 sampling point were employed to carry out the fourier transform using Padé approximants. Violet line: 8001 sampling point were used to carry out the standard Fourier Transform.

3 Simple Python function *get_Fock*(D)

```
def get_Fock(D, mints, f_type, basisset):
    mints = psi4.core.MintsHelper(basisset)
    #get ERI
    I=np.array(mints.ao_eri())
    T=np.array(mints.ao_kinetic())
    V=np.array(mints.ao_potential())
    Hcore=T+V
    # Build J,K matrices
    J = np.einsum('pqrs,rs->pq', I, D)
    if (f_type=='hf'):
        K = np.einsum('prqs,rs->pq', I, D)
        F = Hcore + J*np.float_(2.0) - K
        Exc=0.0
        J_ene = 0.0
    else:
        #D must be a psi4.core.Matrix object not a numpy.narray
        restricted = True
        build_superfunctional = \
             psi4.driver.dft_funcs.build_superfunctional
        sup = build_superfunctional(f_type, restricted)[0]
        sup.set_deriv(2)
        sup.allocate()
        vname = "RV"
        if not restricted:
            vname = "UV"
```

```
    potential=psi4.core.VBase.build(basisset,sup,vname)
    Dm=psi4.core.Matrix.from_array(D.real)
    potential.initialize()
    potential.set_D([Dm])
    nbf=D.shape[0]
    V=psi4.core.Matrix(nbf,nbf)
    potential.compute_V([V])
    potential.finalize()
    F = Hcore + J*np.float_(2.0) +V
    Exc= potential.quadrature_values()["FUNCTIONAL"]
    if sup.is_x_hybrid():
      alpha = sup.x_alpha()
      K = np.einsum('prqs,rs->pq', I, D)
      F += -alpha*K
      Exc += -alpha*np.trace(np.matmul(D,K))
    J_ene=2.00*np.trace(np.matmul(D,J))
return J_ene,Exc,F
```

4 Comparison of our new RT-TDKS (and RT-TDDKS) implementation with Psi4NumPy (with PyBERTHA) and Linear-Responce from NWChem.

In table S1 we are reporting the principal excitation frequency obtained from our new RT-TDKS with Psi4NumPy code and using Linear-Responce in NWChem for the water molecule. The reported results clearly show the correctness of our implementation.

Table S1: Excitation frequency corresponding to the few low-lying transitions are obtained from our new RT-TDKS with Psi4NumPy code and Linear-Response (LR) calculations from NWChem respectively, at BLYP/Def2-Xvp (X = s,tz) level of theory

Excitation energy (e.V)			
Def2-svp		Def2-tzvp	
Psi4NumPy	LR	Psi4NumPy	LR
9.28	9.28	9.16	9.16
16.51	16.51	15.47	15.47
25.88	25.88	18.40	18.40
25.97	25.97	19.42	19.41

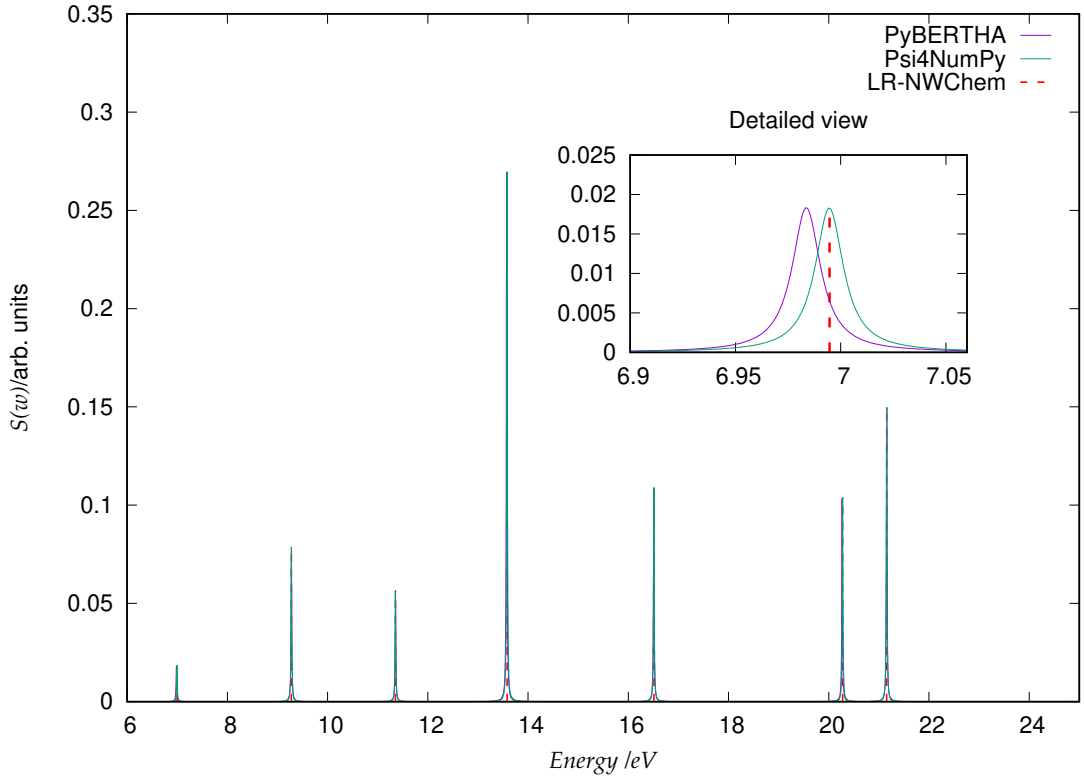

Figure S2: Four component RT-TDDKS absorption spectrum of water molecule (violet line) at BLYP/Def2-svp level is shown. The RT-TDKS absorption spectrum (green line) obtained our new Psi4NumPy implementation is reported for comparison. In the inset a detailed view of HOMO-LUMO transition has been reported and compared with the excitation energy from LR (NWChem) calculation (red dashed line).

5 Absorption spectra of water using RT-TDDKS at BLYP/Def2-tz2p level using different auxiliary basis set.

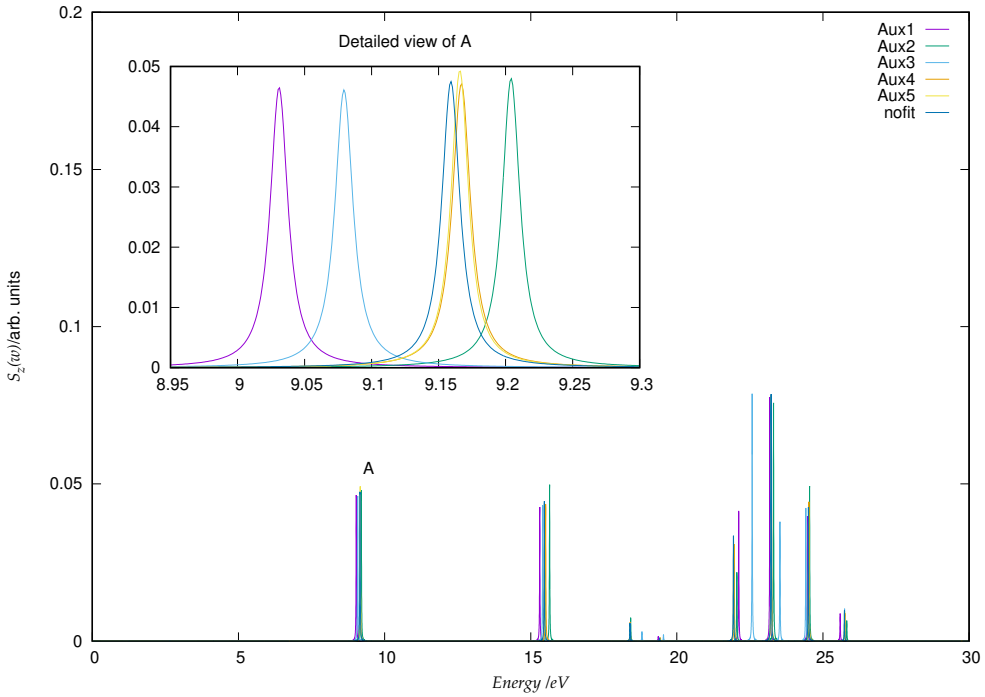

Figure S3: Four component RT-TDDKS dipole strenght function (S_z) of water molecule at BLYP/Def2-tzvp level of theory, for different auxiliary sets. The absorption profile from Four component RT-TDDKS without the density fitting is also reported (indigo line).

6 Vertical excitations in the Zn, Cd, Hg atoms (0-20 eV).

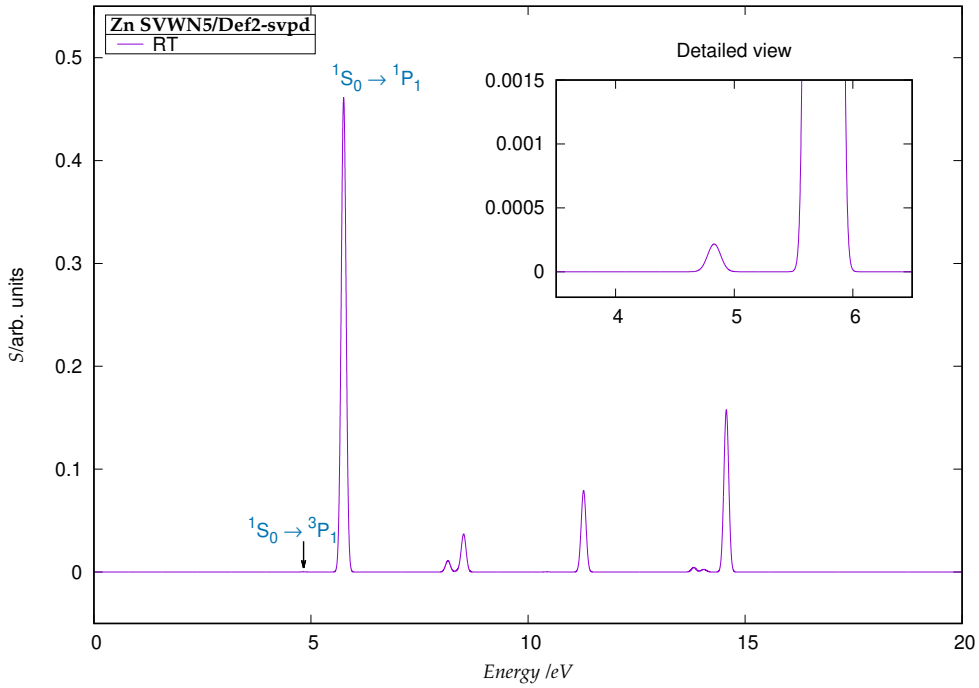

Figure S4: RT-TDDKS absorption spectrum of atomic Zinc in the range 0-20 eV.

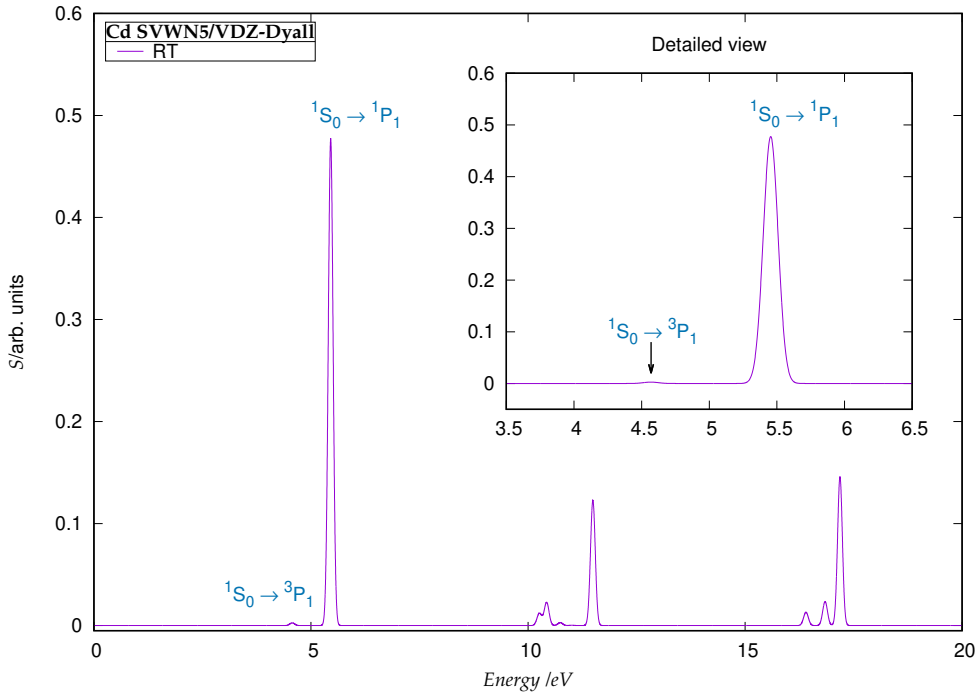

Figure S5: RT-TDDKS absorption spectrum of atomic Cadium in the range 0-20 eV.

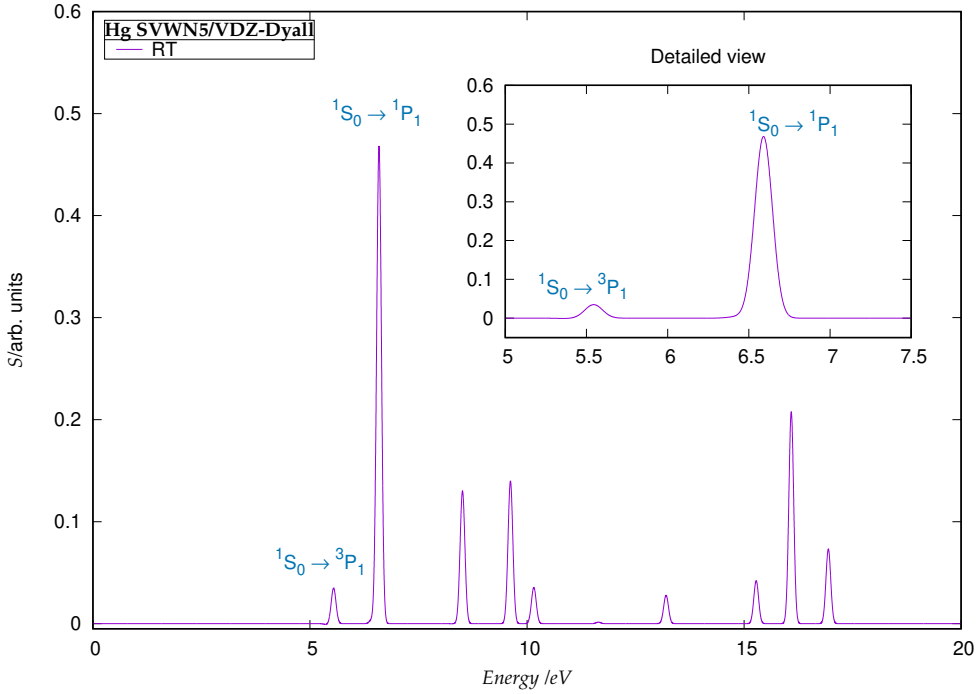

Figure S6: RT-TDDKS absorption spectrum of atomic Mercury in the range 0-20 eV.

7 Intesity of spin-forbidden transition ($^1S_0 \rightarrow {}^3P_1$) increases as Z^5 in the group 12 atoms

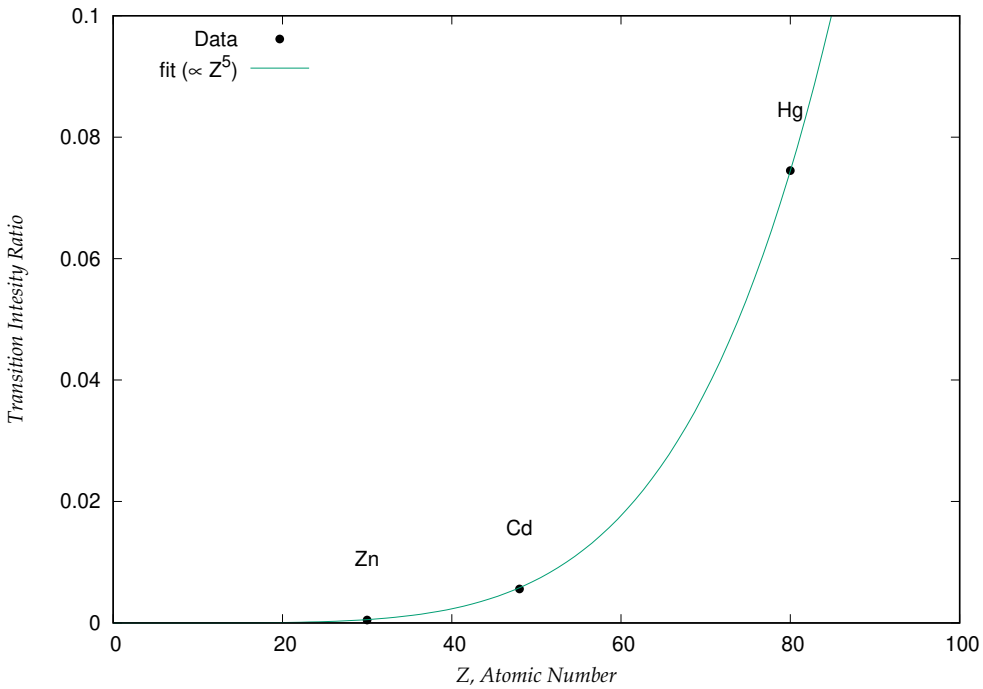

Figure S7: Intesity of the spin-forbidden transition ($^1S_0 \rightarrow {}^3P_1$) as a function of the atomic number, Z.

8 Geometry and density fitting basis sets for water molecule.

Geometry in XYZ format (Å)

```
O 0.000000 -0.000014 -0.348240
H 0.000000  0.760011 -0.932852
H 0.000000 -0.759996 -0.932908
```

Density fitting basis sets

```
Aux 1


H
4
 0.45000000E+02   0
 0.75000000E+01   0
 0.30000000E+00   0
 0.15000000E+01   1


O
7
   2000.00000000  0
    400.00000000  0
    100.00000000  0
     25.00000000  0
      7.80000000  2
      1.56000000  2
      0.39000000  2
-------------------
Aux 2
```

```
H
4
 0.45000000E+02   2
 0.75000000E+01   2
 0.30000000E+00   2
 0.15000000E+01   3

O
7
   2000.00000000  2
    400.00000000  2
    100.00000000  2
     25.00000000  2
      7.80000000  4
      1.56000000  4
      0.39000000  4
-------------------
Aux 3

H
4
  0.45000000E+02   0
  0.75000000E+01   0
  0.30000000E+00   0
  0.15000000E+01   2
```

```
O
8
   2566.00000000   0
    570.00000000   0
    163.00000000   0
     46.50000000   0
     17.00000000   2
      3.80000000   2
      1.08000000   2
      0.31000000   2
-------------------
Aux 4

H
4
 0.45000000E+02   2
 0.75000000E+01   2
 0.30000000E+00   2
 0.15000000E+01   4

O
8
  2566.00000000   2
   570.00000000   2
   163.00000000   2
    46.50000000   2
    17.00000000   4
```

```
     3.80000000   4
     1.08000000   4
     0.31000000   4
------------------
Aux 5

H
6
     37.488614   2
      9.094043   2
      2.767778   2
      1.029648   4
      0.442000   4
      0.200000   2

O
12
   2664.478091   2
    983.266655   2
    377.953261   2
    150.898437   2
     62.375010   2
     26.597608   2
     11.653067   2
      5.222847   4
      2.383464   4
      1.102031   4
```

```
0.513600   4

0.240000   4
```