Journal Name

ARTICLE TYPE

Cite this: DOI: 10.1039/xxxxxxxxxx

# Highly accurate computations to simulate phosphorescence spectra of large transition complexes: a new way to predict color†

Adrien Stoliaroff,[a] Jérémy Rio,[a] and Camille Latouche*[a]

Received Date
Accepted Date

DOI: 10.1039/xxxxxxxxxx

www.rsc.org/journalname

[a] *Institut des Materiaux Jean Rouxel (IMN), Université de Nantes, CNRS, 2 rue de la Houssinière, BP 32229, 44322 Nantes cedex 3, France, E-mail: camille.latouche@univ-nantes.fr*

† Electronic Supplementary Information (ESI) available: [details of any supplementary information available should be included here]. See DOI: 10.1039/b000000x/

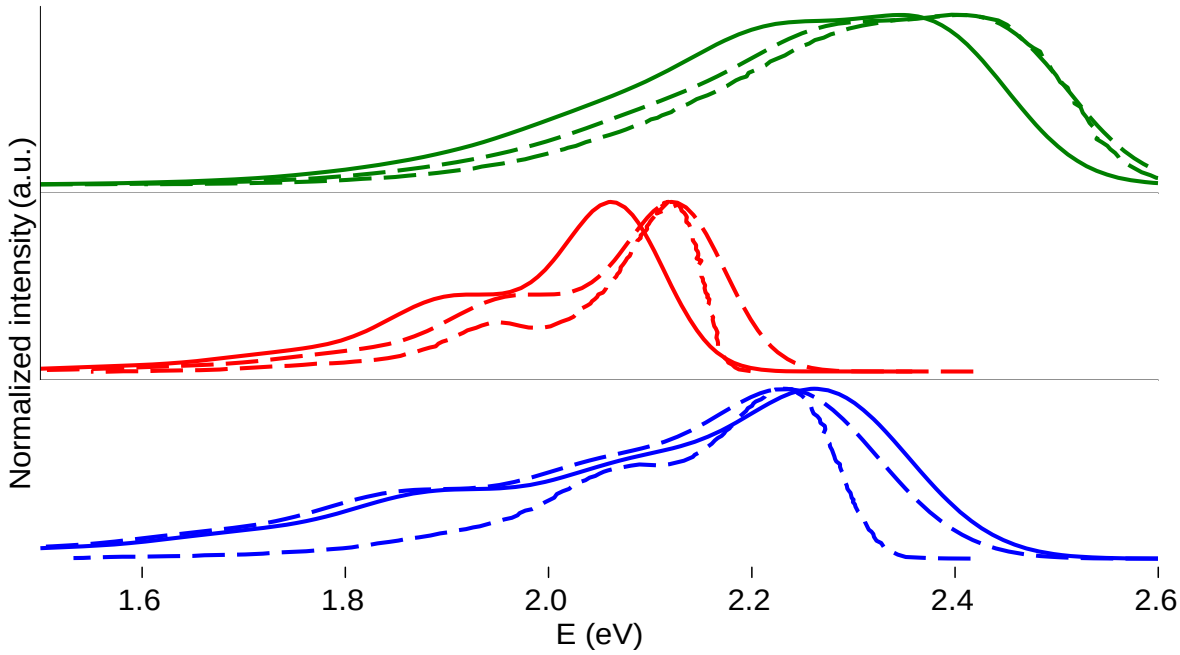


**Fig. 1** Simulated[AH] (plain), shifted (dotted) and recorded [1] (dashed) luminescence spectra of $Ir(ppy)_3$ (green), complex **1** (red) and complex **2** (blue).

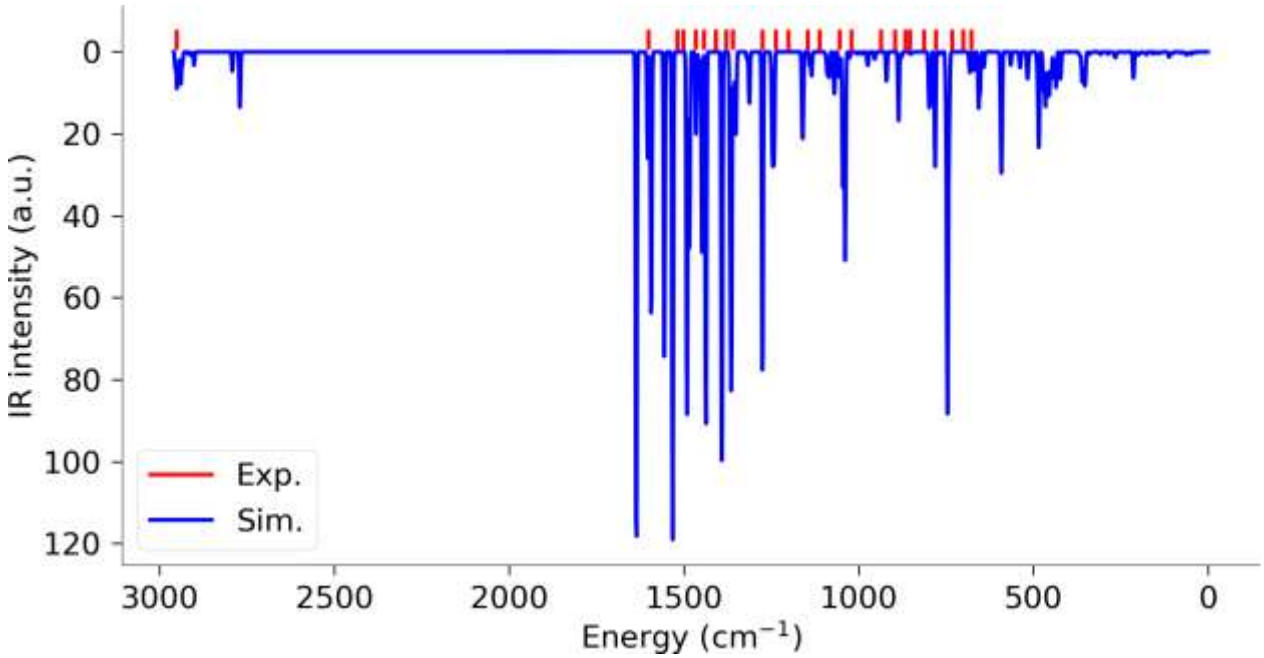


**Fig. 2** Simulated (blue) and experimental [1] (red) infrared transmittance spectra of complex **1**. A correction factor of 0.93 was applied to simulated energies above 2000 cm$^{-1}$.

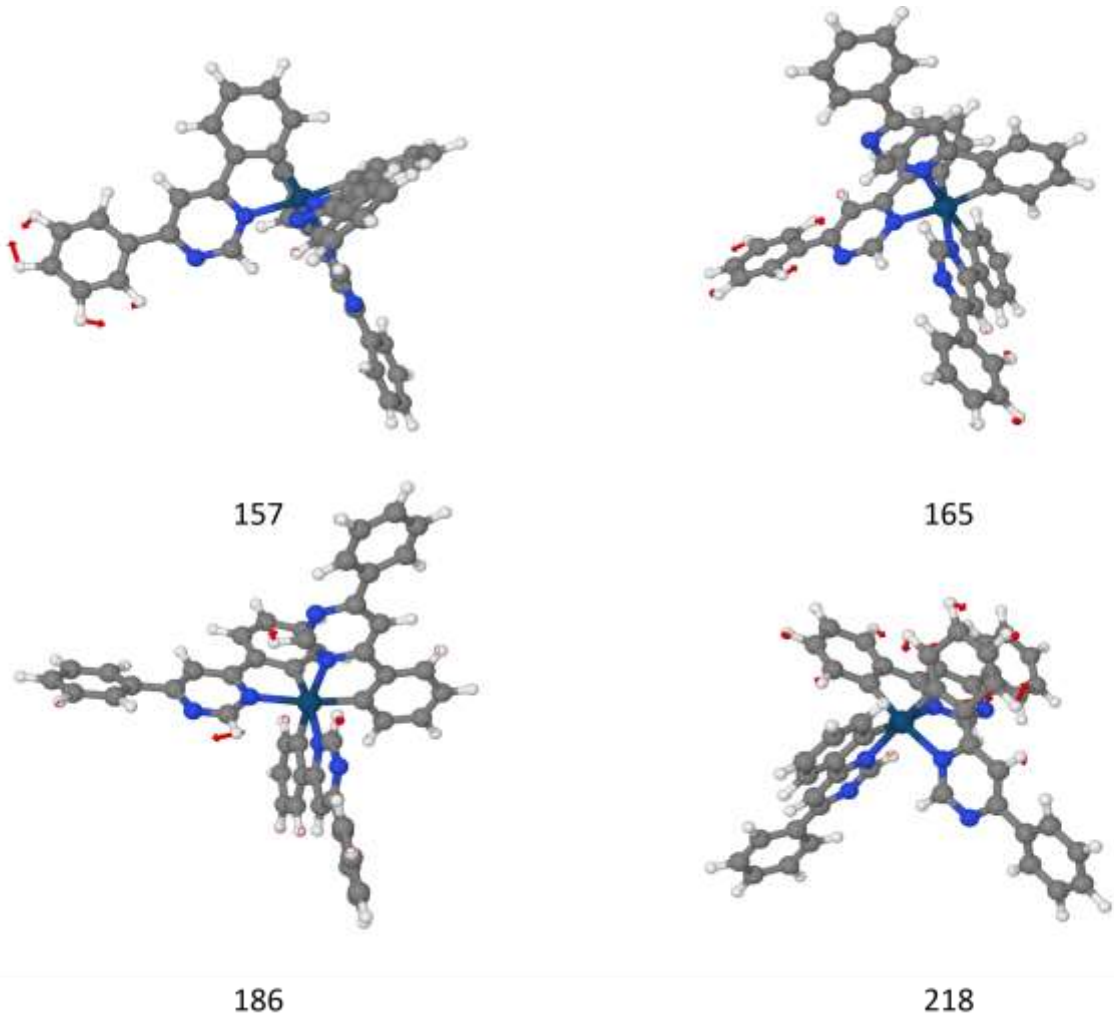


**Fig. 3** Main vibrational modes of complex **2**.

```
# -*- coding: utf-8 -*-

"""
CIE Chromaticity Diagrams Plotting
==================================
Necessary package: colour-science

via pip:
pip install colour-science

or

https://pypi.org/project/colour-science/h
ttps://github.com/colour-science/colour/"
"""

import matplotlib.pyplot as plt
import colour
from colour.plotting import *
import numpy as np
import pandas as pd
import pyvalence_kernel.basic_functions as bf


direc = '' # YOUR PATH
filename = '' # YOUR FILENAME

#############################################################################
#                                AUXILIARY FUNCTIONS
#############################################################################
# CIE1931
cie = np.loadtxt('./CIE1931.csv',delimiter=',') # LOAD CIE BASE FUNCTIONS

# Linear interpolation
def lin_int(grid,fpoints,x):
    try:
        return fpoints[list(grid).index(x)]
    except ValueError,e:
        try:
            k = [x <= xk for xk in grid].index(True) # xk-1<x<xk
        except ValueError,e:
            return 0
        if (k > 0) and k<len(grid):
            return fpoints[k-1]+(x-grid[k-1])*(fpoints[k]-fpoints[k-1])/(grid[k]-grid[k-1])
        else:
            return 0

# Indicator function
def ind(x0,x1,x):
    if (x0<=x) and (x<=x1):
        return True
    else:
        return False

#############################################################################
#                                MAIN FUNCTION
```

```
#########################################################################
def spectrum_to_cie(simulation , gridheader , spectrheader , energy=True , plot=False , wlmin=400,wlmax=700):
    """
    Calculates the CIE coordinates from the given spectrum using Beck's procedure doi.org/10.1002/qua.2032
    Parameters
    ----------
    simulation : spectrum as a pandas dataframe.
    gridheader : wavelength/energy header as a string.
    spectrheader : intensity header as a string.
    energy : True if intensity vs energy, False if vs wavelength.
    plot : boolean, if True display spectrum.
    wlmin : minimum of wavelength range for integration.
    wlmin : maximum of wavelength range for integration.
    -------
    tuple
        X,Y,Z in CIE coordinates.
    Example
    --------
    >>> spectrum_to_cie(exptemp1, 'lambda','I', False, False, 550, 800)
    """
    if energyis True:
        simulation = simulation.sort_values(by=0,ascending=False)
        simulation = simulation.reset_index(drop=True)
        wavegrid = (1240/simulation[gridheader]) # wavelengthgrid
    else:
        wavegrid = simulation[gridheader]
    dlambda = wavegrid[1:]-wavegrid[:-1]
    norm_spectr = simulation[spectrheader]/max(simulation[spectrheader])
    treated_spectr = max(norm_spectr)-min(norm_spectr)-norm_spectr
    if plotis True:
        labels = ['X','Y','Z']
        for i in range(1,4):
            fig = plt.figure()
            ax1 = fig.add_subplot(311)
            ax2 = fig.add_subplot(312)
            ax3 = fig.add_subplot(313)
            fig.show()
            ax1.set_title('%s( CIE)'%labels[i-1],fontsize=30)
            ax2.set_title('sim_spectrum',fontsize=30)
            ax3.set_title('%s* spectr'%labels[i-1],fontsize=30)
            ax1.plot(wavegrid,[lin_int(cie[:,0],cie[:,i],dwave) for dwave in wavegrid],lw=3)
            ax2.plot(wavegrid,[lin_int(wavegrid,norm_spectr,dwave) for dwave in wavegrid],"--",lw=3)
            ax2.plot(wavegrid,[lin_int(wavegrid,treated_spectr,dwave) for dwave in wavegrid],lw=3)
            ax3.plot(wavegrid, [lin_int(cie[:,0],cie[:,i],dwave)* lin_int(wavegrid,treated_spectr,dwave) fo
            for ax in [ax1,ax2,ax3]:
                ax.set_xlim(wlmin,wlmax)
    return [sum([ind(wlmin,wlmax,dwave)* lin_int(cie[:,0],cie[:,i],dwave)* lin_int(wavegrid,treated_spectr,

#########################################################################
#                          PLOT IN CIE COORDINATES
#########################################################################

# Read spectrum CSV
spectrum = pd.read_csv(direc+filename,header=None,delim_whitespace=True)
# Plot the * CIE 1931 Chromaticity Diagram * .
xex,yex,zex = spectrum_to_cie(spectrum,0,1,False,False,550,800) # integrate intensity from 550 nm t
```

```
fig, ax = plot_chromaticity_diagram_CIE1931(standalone=False, show_spectral_locus=True)
ax.plot(xex/(xex+yex+zex),yex/(xex+yex+zex),'o-',color='black')

fig.show()
```

## Notes and references

1G. Turnbull, J. A. G. Williams and V. N. Kozhevnikov, *Chemical Communications*, 2017, **53**, 2729–2732.