pgn_plot

November 8, 2022

[3]:
```
# PROTEIN GRAPH NETWORK: MAKE PLOT
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as colors
from mpl_toolkits import mplot3d

# helper function to color atoms following CPK coloring rules
def atom_color(item):
    item = str(item).title()
    if item=='H':
        color = 'silver'
    elif item=='C':
        color = 'black'
    elif item=='N':
        color = 'blue'
    elif item=='O':
        color = 'red'
    elif item=='F' or item=='Cl':
        color = 'lime'
    elif item=='Br':
        color = 'maroon'
    elif item=='I':
        color = 'purple'
    elif item=='He' or item=='Ne' or item=='Ar' or item=='Kr' or item=='Xe':
        color = 'cyan'
    elif item=='P':
        color = 'orange'
    elif item=='S':
        color = 'yellow'
    elif item=='B':
        color = 'lightsalmon'
    elif item=='Li' or item=='Na' or item=='K' or item=='Rb' or item=='Cs' or␣
 ↪item=='Fr':
        color = 'darkviolet'
    elif item=='Be' or item=='Mg' or item=='Ca' or item=='Sr' or item=='Ba' or␣
 ↪item=='Ra':
        color = 'green'
```

```
    elif item=='Ti':
        color = 'gray'
    elif item=='Fe':
        color = 'chocolate'
    else:
        color = 'pink'
    return colors.to_rgba(color)

# helper function to color bonds
def bond_color(item):
    item = str(item).upper()
    if item=='BCS':
        color = 'silver'
    elif item=='BCD':
        color = 'gray'
    elif item=='RCS':
        color = 'turquoise'
    elif item=='RCD':
        color = 'teal'
    elif item=='HCS':
        color = 'deeppink'
    elif item=='HCD':
        color = 'purple'
    elif item=='HCT':
        color = 'indigo'
    elif item=='HHH':
        color = 'orange'
    elif item=='HPO':
        color = 'wheat'
    else:
        color = 'lime'
    return colors.to_rgba(color)

# helper function to plot atoms
def plot_atoms(ax,atom_list,atom_params,plot_params):

    atom_size = float(atom_params[0]) # additional plot parameters
    atom_style = str(atom_params[1])

    vec_atom_color = np.vectorize(atom_color) # vectorize coloring function
    if plot_params[0]==True: # plot_backbone_atoms
        ib = atom_list[:,3]==['B']
        try:
            color = vec_atom_color(atom_list[ib,10]) # select only backbone␣
 ↪rows with boolean mask
```

```
            ax.scatter(atom_list[ib,7],atom_list[ib,8],atom_list[ib,9],c=np.
↪transpose(color),marker=atom_style,s=atom_size,zorder=2) # plot backbone␣
↪points
        except ValueError:
            print('WARNING: no backbone atoms identified')
    if plot_params[1]==True: # plot_residue_atoms
        ir = (atom_list[:,3])==['R']
        try:
            color = vec_atom_color(atom_list[ir,10]) # mask only residue rows␣
↪with boolean mask
            ax.scatter(atom_list[ir,7],atom_list[ir,8],atom_list[ir,9],c=np.
↪transpose(color),marker=atom_style,s=atom_size,zorder=2) # plot residue␣
↪points
        except ValueError:
            print('WARNING: no residue atoms identified')
    if plot_params[2]==True: # plot_hetatm_atoms
        ih = (atom_list[:,3])==['H']
        try:
            color = vec_atom_color(atom_list[ih,10]) # select only hetatm rows␣
↪with boolean mask
            ax.scatter(atom_list[ih,7],atom_list[ih,8],atom_list[ih,9],c=np.
↪transpose(color),marker=atom_style,s=atom_size,zorder=2) # plot hetatm points
        except ValueError:
            print('WARNING: no hetatm atoms identified; check that atom array␣
↪parameters include desired hetatm names')
    if plot_params[3]==True: # plot_water_atoms
        iw = (atom_list[:,3])==['W']
        try:
            color = vec_atom_color(atom_list[iw,10]) # select only water rows␣
↪with boolean mask
            ax.scatter(atom_list[iw,7],atom_list[iw,8],atom_list[iw,9],c=np.
↪transpose(color),marker=atom_style,s=atom_size,zorder=2) # plot water points
        except ValueError:
            print('WARNING: no water atoms identified')

# helper function to plot bonds
def plot_bonds(ax,atom_list,bond_list,bond_params):

    bond_linewidth = float(bond_params[0]) # additional plot parameters
    bond_style = str(bond_params[1])

    for bond in bond_list:
        i = bond[1]
        j = bond[2]
        c = bond_color(bond[3])
```

```
        i_x = atom_list[i,7]
        i_y = atom_list[i,8]
        i_z = atom_list[i,9]

        j_x = atom_list[j,7]
        j_y = atom_list[j,8]
        j_z = atom_list[j,9]

        ax.
 ↪plot([i_x,j_x],[i_y,j_y],[i_z,j_z],color=c,ls=bond_style,lw=bond_linewidth,zorder=1)

def make_plot(bond_list,atom_list,atom_params,bond_params,plot_params):
    print('\n[running make_plot]')

    plt.close('all')
    fig = plt.figure()
    ax = plt.axes(projection='3d',computed_zorder=False) # zorder used to set␣
 ↪plotting order

    plot_bonds(ax,atom_list,bond_list,bond_params)
    plot_atoms(ax,atom_list,atom_params,plot_params)

    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_zlabel('z')

    mins = np.amin(atom_list[:,7:10],axis=0) # scale plot evenly on all 3 axes
    maxes = np.amax(atom_list[:,7:10],axis=0)
    avg = np.average([mins,maxes],axis=0)
    diff = 0.5*np.amax((maxes-mins))
    ax_mins = avg-diff
    ax_maxes = avg+diff

    ax.set_xlim([ax_mins[0],ax_maxes[0]])
    ax.set_ylim([ax_mins[1],ax_maxes[1]])
    ax.set_zlim([ax_mins[2],ax_maxes[2]])
```

[ ]: