charge_transport_demo

May 25, 2022

[ ]:
```
import sys
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import collections as mc
import os
import pathlib
import random
import importlib as imp
import scipy.stats as stats
import scipy.optimize as opt
import itertools as iter
import math
import ipywidgets
from tqdm.notebook import tqdm, trange
import datetime
import time
import json
import h5py
import lmfit
import pandas as pd
import datetime
import psutil

import flow_fields as flow
import charge_transfer_assets as assets
for mod in [flow, assets]:
    imp.reload(mod)

# Set matplotlib format
SMALL_SIZE = 12
MEDIUM_SIZE = 15
BIGGER_SIZE = 18

plt.rc('font', size=SMALL_SIZE)          # controls default text sizes
plt.rc('axes', titlesize=SMALL_SIZE)     # fontsize of the axes title
plt.rc('axes', labelsize=MEDIUM_SIZE)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
```

```
plt.rc('ytick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('legend', fontsize=SMALL_SIZE)    # legend fontsize
plt.rc('figure', titlesize=BIGGER_SIZE)  # fontsize of the figure title
```

[ ]:
```
print('RAM memory % used:', psutil.virtual_memory()[2])
print('RAM memory Available: %.1f GB' % (int(psutil.virtual_memory()[1]) / 10.
↪0**9))
```

1 Run Simulations

[ ]:
```
for mod in [flow, assets]:
    imp.reload(mod)
##########################################################################################
# Chain Variables
##########################################################################################

xyz_path = '../data/chain_geometries'
files = np.array([f for f in os.listdir(xyz_path) if 'r50v' in f])
field_strengths = [int(f.split('v')[1]) for f in files]

sorted_files = files[np.argsort(field_strengths)]
sorted_field_strengths = np.sort(field_strengths)

step_size = 10  # Real space distance between 4D-STEM diffraction patterns, in␣
↪nm
chain_length = 54  # Length of polymer chains, in nm

shape = (80, 80, 1)  # Number of bins in each (row, column, pillar). Creates␣
↪default settings for analysis.

##########################################################################################
# Charge Transfer Variables
##########################################################################################

# Compute Rates
n_times = 10  # Time resolution of probability densities.  Should be >= 5.

# Propagate Charges
n_charges = 100  # Number of charges in each starting bin.  Total charges =␣
↪n_charges * n_rows * (start_width/step_size)
t_min = 10**-6  # Time of first save point in seconds
t_max = 10**-1  # Time of last save point in seconds
linear_spacing = False  # False is default - logarithmic spacing between save␣
↪points
```

```python
start_padding = 0  # distance between near wall of simulation and charge␣
 ↪starting point, in nm
end_padding = 50 # width of absorbing boundary, in nm
start_width = 10 # width of starting band for charges, in nm
ct_bin_size = 5 # Maximum distance between two "neighbors" for interchain␣
 ↪transport, in nm
n_blocks = 201 # number of save points
energetic_disorder = 0.00  # Standard deviation of random potential on each␣
 ↪bead, in eV
track_single_moves = True  # Record the position of charges after each move. ␣
 ↪Increases RAM use by ~1-10GB.
save_big_arrays = False  # If true, save pre-tabluated rate matrices and data␣
 ↪on individual moves.  Adds ~1-10GB per simulation.

##########################################################################################
# Output Variables
##########################################################################################
output_dir = '../data/transport_simulations/demo_code/'  # Dir must be created␣
 ↪manually
batch_name = ''


##########################################################################################
# Batch Values
##########################################################################################

input_files = [10]  # Different levels of alignment
field_values = [10**7, 10**6]  # V/m
angle_values = [0, 90] # any angle for short-range transport information.  0 or␣
 ↪90 for long-range transport
sign_values = [1]  # Direction of charges
case_values = [0]  # Case does nothing by default, but is a way for the user to␣
 ↪define new settings easily
skip_duplicates = False  # if True, code will skip simulations that have␣
 ↪already been run (or partially run).

print('Total Simulations:', len(input_files) * len(field_values) *␣
 ↪len(angle_values) * len(sign_values))

##########################################################################################
# Special Experiments
##########################################################################################

shuffle = False
align = False
make_rigid = False
```

```python
modify_fraction = 1.0

###############################################################################################
# Check initial performance
###############################################################################################


print('\n\nInitial Performance:')
print('RAM memory % used:', psutil.virtual_memory()[2])
print('RAM memory Available: %.1f GB' % (int(psutil.virtual_memory()[1]) / 10.0**9))
print('CPU Used: %.0f percent\n\n' % psutil.cpu_percent(5))

###############################################################################################
# Check Existing Files
###############################################################################################


summary_strings = []

for i in os.walk(output_dir):
    data_folders = [name for name in i[1] if '2022' in name]
    break
else:
    data_folders = []
for index, name in enumerate(data_folders):
    with open(output_dir + name + '/simple_attributes.txt') as f:
        data = json.load(f)
        field = abs(data['field'])
        sign = data['sign']
        angle = data['angle']
    with open(output_dir + name + '/chains_simple_attributes.txt') as f:
        data = json.load(f)
        file_number = int(data['xyz_full_path'].split('v')[-1]) - 1
    summary = '%d %.0f %.0f %.0f' % (file_number, field, angle, sign)
    summary_strings.append(summary)

print('Existing Files:')
summary_strings.sort()
for i, name in enumerate(summary_strings):
    print(i, name)
print('\n')
print('Total Unique Files: %d\n\n' % len(set(summary_strings)))


###############################################################################################
# Main Loop
```

```
################################################################################

print('Files in Loop:')
counter = 0

for n, case in enumerate(case_values):
    for i, file_number in enumerate(input_files):
        for j, field in enumerate(field_values):
            for k, angle in enumerate(angle_values):
                for m, sign in enumerate(sign_values):

                    summary = '%d %.0f %.0f %.0f' % (file_number, field, angle,␣
 ↪sign)
                    print(counter, summary)
                    if summary in summary_strings:
                        if skip_duplicates:
                            counter += 1
                            continue
                    counter += 1


                    xyz, xyz_full_path = assets.get_xyz(xyz_path, sorted_files,␣
 ↪file_number, step_size, rotate=True)
                    chain_data = assets.ChainSet(xyz, chain_length,␣
 ↪xyz_full_path,
                                                 shuffle=shuffle, align=align,␣
 ↪make_rigid=make_rigid, modify_fraction=modify_fraction)
                    chain_data.create_bins(shape)

                    exp_1 = assets.
 ↪PathExperiment(save_big_arrays=save_big_arrays)

                    exp_1.compute_rates(chain_data, field * sign, angle,␣
 ↪verbose=False, images=True, n_times=n_times,␣
 ↪energetic_disorder=energetic_disorder)

                    exp_1.propagate_charges(n_charges, t_min=t_min,␣
 ↪t_max=t_max, linear_spacing=linear_spacing, verbose=False,
                                            start_padding=start_padding,␣
 ↪end_padding=end_padding, start_width=start_width,
                                            bin_size=ct_bin_size,␣
 ↪n_blocks=n_blocks, track_single_moves=track_single_moves)
                    print('Unpack Results...')
                    exp_1.unpack_results()
                    print('Save Results...')
                    exp_1.to_file(output_dir, batch_name=batch_name)
```

2 Analysis

[ ]:
```
file_manager = assets.FileManager('../data/transport_simulations/')
file_manager.add_files('ct_29_n100/', angle=90, field=10**6)
```

[ ]:
```
for f in file_manager.files:
    f.unpack_results()
    f.make_default_figures()
```

[ ]:
```
```