NetSci Documentation

1 Installation

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

```
chmod +x Miniconda3-latest-Linux-x86_64.sh
```

```
source ~/.bashrc
```

```
conda install -c conda-forge git
```

```
git clone https://github.com/amstokely/netsci.git
```

```
cd netsci
```

```
NETSCI_ROOT=$(pwd)
```

```
conda env create -f netsci.yml
```

```
source activate netsci
```

```
mkdir ${NETSCI_ROOT}/build
```

```
cd ${NETSCI_ROOT}/build
```

```
nvcc ${NETSCI_ROOT}/build_scripts/cuda_architecture.cu -o cuda_architecture
```

```
CUDA_ARCHITECTURE=$(./cuda_architecture)
```

```
cmake .. -DCONDA_DIR=$CONDA_PREFIX -DCUDA_ARCHITECTURE=${CUDA_ARCHITECTURE}
```

```
cmake --build . -j
```

```
make python
```

```
ctest
```

```
cd ${NETSCI_ROOT}/tests/cuarray/python
```

```
pytest
```

```
cd ${NETSCI_ROOT}/tests/netcalc/python
```

```
pytest
```

2 CuArray Tutorial

```
import os

import numpy as np

import cuarray
```

```
cu_a = cuarray.FloatCuArray()
```

```
cu_a.init(8, 5)
```

```
m, n = cu_a.m(), cu_a.n()
```

```
for i in range(m):
    for j in range(n):
        print(f'{cu_a[i][j]:.4f}', end=' ')
    print()
```

```
for i in range(m):
    for j in range(n):
        cu_a[i][j] = np.random.random()
```

```
for i in range(len(cu_a)):
    for j in range(len(cu_a[0])):
        print(f'{cu_a[i][j]:.4f}', end=' ')
    print()
```

```
cu_a_row0_sorted = cu_a.sort(0)
```

```
for i in range(len(cu_a_row0_sorted[0])):
    print(f'{cu_a_row0_sorted[0][i]:.4f}', end=' ')
```

```
cu_a_row0_sorted_indices = cu_a.argsort(0)
```

```
for i in range(len(cu_a_row0_sorted_indices[0])):
    sorted_index = cu_a_row0_sorted_indices[0][i]
    print(f'{sorted_index}    {cu_a[0][sorted_index]:.4f}', end='\n')
```

```
cu_a_sorted = cuarray.FloatCuArray()
cu_a_sorted.init(cu_a.m(), cu_a.n())
```

```
for i in range(len(cu_a)):
    cu_a_sorted[i] = cu_a.sort(i)[0]
```

```
for i in range(len(cu_a)):
    for j in range(len(cu_a[0])):
        print(f'{cu_a_sorted[i][j]:.4f}', end=' ')
    print()
```

```
cu_a_sorted_indices = cuarray.IntCuArray()
cu_a_sorted_indices.init(cu_a.m(), cu_a.n())
```

```
for i in range(len(cu_a)):
    cu_a_sorted_indices[i] = cu_a.argsort(i)[0]
```

```
for i in range(len(cu_a)):
    for j in range(len(cu_a[0])):
        sorted_index = cu_a_sorted_indices[i][j]
        print(f'{cu_a_sorted[i][j]:.4f}', end=' ')
    print()
```

```
cu_a_fname = f'{os.getcwd()}/cu_a.npy'
```

```
cu_a.save(cu_a_fname)
```

```
cu_a_from_load = cuarray.FloatCuArray()
```

```
cu_a_from_load.load(cu_a_fname)
```

```
np_a = cu_a.toNumpy2D()
```

```
for i in range(m):
    for j in range(n):
        print(f'{cu_a[i][j]:.4f}', end=' ')
    print(end='    ')
    for j in range(n):
        print(f'{np_a[i][j]:.4f}', end=' ')
    print()
```

```
np_a_linear = cu_a.toNumpy1D()
```

```
for i in range(m):
    for j in range(n):
        print(f'{cu_a[i][j]:.4f}', end=' ')
    print(end='    ')
    for j in range(n):
        linear_index = i*n + j
        print(f'{np_a_linear[linear_index]:.4f}', end=' ')
    print()
```

```
cu_a_np = cuarray.FloatCuArray()
cu_a_np.fromNumpy2D(np_a)
```

```
for i in range(m):
    for j in range(n):
        print(f'{cu_a[i][j]:.4f}', end=' ')
    print(end='    ')
    for j in range(n):
        print(f'{cu_a_np[i][j]:.4f}', end=' ')
    print()
```

```
cu_a_np_linear = cuarray.FloatCuArray()
cu_a_np_linear.fromNumpy1D(np_a_linear)
```

```
for i in range(m):
    for j in range(n):
        linear_index = i*n + j
        print(f'{cu_a_np_linear[0][linear_index]:.4f}', end=' ')
    print(end='    ')
    for j in range(n):
        linear_index = i*n + j
        print(f'{np_a_linear[linear_index]:.4f}', end=' ')
    print()
```

```
cu_a_copy = cuarray.FloatCuArray()
cu_a_copy.fromCuArray(
    cuArray=cu_a,
    start=0,
    end=len(cu_a)-1,
    m=cu_a.m(),
    n=cu_a.n(),
)
```

```
for i in range(len(cu_a)):
    for j in range(len(cu_a[0])):
        print(f'{cu_a_copy[i][j]:.4f}', end=' ')
    print(end='    ')
    for j in range(len(cu_a[0])):
        print(f'{cu_a[i][j]:.4f}', end=' ')
    print()
```

```
cu_a_from_linear = cuarray.FloatCuArray()
cu_a_from_linear.fromCuArray(
    cuArray=cu_a_np_linear,
    start=0,
    end=0,
    m=cu_a.m(),
    n=cu_a.n(),
)
```

```
for i in range(len(cu_a)):
    for j in range(len(cu_a[0])):
        print(f'{cu_a_from_linear[i][j]:.4f}', end=' ')
    print(end='    ')
    for j in range(len(cu_a[0])):
        print(f'{cu_a[i][j]:.4f}', end=' ')
    print()
```

3 NetChem Tutorial

```
import os
import tarfile

import netchem
```

```
# NETSCI_ROOT=<path to netsci root directory>
tutorial_directory = f'{NETSCI_ROOT}/tutorial'
os.chdir(tutorial_directory)
```

```
tutorial_tarball = tarfile.open(f'{os.getcwd()}/tutorial.tar.gz')
tutorial_tarball.extractall()
tutorial_tarball.close()
```

```
first_frame = 0
last_frame = 999
trajectory_file = f'{tutorial_directory}/pyro.dcd'
topology_file = f'{tutorial_directory}/pyro.pdb'
```

```
graph = netchem.Graph()
```

```
graph.init(
    trajectoryFile=trajectory_file,
    topologyFile=topology_file,
    firstFrame=first_frame,
    lastFrame=last_frame,
)
```

```
num_nodes = graph.numNodes()
print(num_nodes)
```

```
num_frames = graph.numFrames()
print(num_frames)
```

```
num_atoms = atoms.numAtoms()
print(num_atoms)
```

```
for atom in atoms:
    print(atom.index())
```

```
atom0 = atoms.at(0)
```

```
atom0_index = atom0.index()
atom0_serial = atom0.serial()
atom0_name = atom0.name()
atom0_element = atom0.element()
atom0_residue_id = atom0.residueId()
atom0_residue_name = atom0.residueName()
atom0_chain_id = atom0.chainId()
atom0_segment_id = atom0.segmentId()
atom0_mass = atom0.mass()
```

```
atom0_properties_str = f'Index: {atom0_index}\n'
atom0_properties_str += f'Serial: {atom0_serial}\n'
atom0_properties_str += f'Name: {atom0_name}\n'
atom0_properties_str += f'Element: {atom0_element}\n'
atom0_properties_str += f'Residue ID: {atom0_residue_id}\n'
atom0_properties_str += f'Residue Name: {atom0_residue_name}\n'
atom0_properties_str += f'Chain ID: {atom0_chain_id}\n'
atom0_properties_str += f'Segment ID: {atom0_segment_id}\n'
atom0_properties_str += f'Mass: {atom0_mass:.4f}'
```

```
print(atom0_properties_str)
```

```
nodes = graph.nodes()
```

```
for node in nodes:
    print(node.index())
```

```
for node in graph:
    print(node.index())
```

```
node0_tag = node0.tag()
```

```
node0_num_atoms = node0.numAtoms()
```

```
node0_index = node0.index()
```

```
node0_total_mass = node0.totalMass()
```

```
node0_atoms = node0.atoms()
```

```
node0_properties_str = f'Index: {node0_index}\n'
node0_properties_str = f'Number of Atoms: {node0_num_atoms}\n'
node0_properties_str += f'Total Mass: {node0_total_mass:.4f}\n'
node0_properties_str += f'Tag: {node0_tag}\n'
node0_properties_str += f'Node 0 Atoms:\n'
for atom in node0_atoms:
    node0_properties_str += f'    Index: {atom.index()}\n'
    node0_properties_str += f'    Residue Name: {atom.residueName()}\n'
    node0_properties_str += f'    Residue ID: {atom.residueId()}\n'
    node0_properties_str += f'    Chain ID: {atom.chainId()}\n'
    node0_properties_str += f'    Segment ID: {atom.segmentId()}\n'
    node0_properties_str += f'    Mass: {atom.mass():.4f}\n\n'
```

```
print(node0_properties_str)
```

```
node_coordinates = graph.nodeCoordinates()
```

```
node0_coordinates = cuarray.FloatCuArray()
node0_coordinates.fromCuArray(
    cuArray=node_coordinates,
    start=0,
    end=0,
    m=3,
    n=num_frames,
)
```

```
for frame in range(num_frames):
    node0_x = node0_coordinates[0][frame]
    node0_y = node0_coordinates[1][frame]
    node0_z = node0_coordinates[2][frame]
    print(
        f'x: {node0_x:.4f}    '
        f'y: {node0_y:.4f}    '
        f'z: {node0_z:.4f}    '
        f'frame: {frame}'
    )
```

```
graph_json = f'{tutorial_directory}/graph.json'
graph_node_coordinates_npy = f'{tutorial_directory}/graph_node_coordinates.npy'
```

```
graph.save(graph_json)
graph.nodeCoordinates().save(graph_node_coordinates_npy)
```