NM^3: Nonlinear MetaMaterial MPI Solver

This is the repository for the source codes and example files used in paper:

High-Performance Large-Scale Simulation of Multi-stable Metastructures

M. Hwang, C. Scalo, and A. F. Arrieta, *To be published*.

Program overview

NM^3 is massively parallel code for solving general multi-stable mechanical metamaterials and metastructures. The code is written in **Fortran** and utilizes **MPI** (Message-passing interface) system calls to execute parallel computation. The code supports both explicit and implicit numerical methods: up to the 4th-order Runge-Kutta (RK) methods are implemented for the explicit solver; and the Newmark-β (NB) method with constant average acceleration is implemented for the implicit solver. Currently, three types of multi-stable systems can be solved with **NM^3**: (1) 1D multi-stable lattice with coupled pendula (discrete sine-Gordon model), (2) 1D lattice with quartic on-site potentials (discerete φ-4 model), and (3) metabeam with a bi-stable microsctructure.

Before running the program

- System prerequisites: **MPI**, **Fortran** complier, **GNU make**, **Python**, **git**, **hdf5**.
- Clone **NM^3** repository.

  > $ git clone https://github.com/wonnie87/NMCube

- Compile and link the programs

  > $ cd NMCube/NB
  > $ make
  > $ cd ../RK
  > $ make

- (Note:) If **GNU make** gives an error saying that it cannot open 'hdf5.mod' and/or find '-lhdf5,' uncomment line 7 and line 9 in **Makefile** file, set the environment variables **PATH_TO_HDF5_HEADER** and **PATH_TO_HDF5_LIB** to the paths where **HDF5** header and libraries are installed, and re-run **make** command.
  - On my Ubuntu subsystem from Windows app store, the headers are located in **/usr/include/openmpi/**, and the libraries are located in **/usr/lib/x86_64-linux-gnu/hdf5/openmpi**.

Running the program

For both NB and RK solvers, run the following command in their respective folders.

> $ make run NP=(number of processes) INP=(input file name)

For example, to run the input file **InpGen_prob4.py** with RK solver:

> $ cd NMCube/RK
> $ make run NP=8 INP=InpGen_prob4.py

If parameters **NP** and **INP** are not specified, **make run** uses **NP**=4 and **INP**=InputFileGeneration.py by default.

Data storage

The simulated outputs are packaged in a **HDF5** file format (.h5). The time displacements, velocities, and accelerations are stored in the datasets **/x**, **/xdot**, and **/xddot**, respectively.

- (Note:) For NB solver, the outputs at each time step are stored in separate datasets. The data structure will be modified to follow that of RK solver in the next revision.

Use **h5dump** or **h5ls** (default command line programs supplied with **hdf5** installation) for quick data retrieval.

For example, to view the entire time displacements of a file **d04_F0.1_8.0Hz.h5**:

> $ h5dump -d "/x" d04_F0.1_8.0Hz.h5

For more involved postprocessing, use more advanced tools, such as **Python**'s **h5py** library. An example **Python** script can be found in **postprocess_example** folder.

File description

```
.
├── LICENSE                     -> MIT License file
├── NB                          -> [Folder] containing NB solver
│   ├── InpGen_NB_prob1.py        -> Example input for a discrete sine-Gordon model
│   ├── InpGen_NB_prob2.py        -> Example input for a discrete phi-4 model
│   ├── InpGen_NB_prob4.py        -> Example input for a bi-stable metabeam
│   ├── InputFileGeneration_NB.py -> Default input file
│   ├── Makefile                  -> Makefile compiling and running the code
│   ├── hdf5_helpers.f90          -> Helper module containing convenient subroutines for handling HDF5 files
│   ├── main_NB.f90               -> Main Fortran script
│   └── sub_NB.f90                -> Fortran module containing subroutines used in main_NB.f90
├── README.md                   -> README file (User guide)
├── RK                          -> [Folder] containing RK solver
│   ├── InpGen_prob1.py           -> Example input for a discrete sine-Gordon model
│   ├── InpGen_prob2.py           -> Example input for a discrete phi-4 model
│   ├── InpGen_prob4.py           -> Example input for a bi-stable metabeam
│   ├── InputFileGeneration.py    -> Default input file
│   ├── Makefile                  -> Makefile compiling and running the code
│   ├── hdf5_helpers.f90          -> Helper module containing convenient subroutines for handling HDF5 files
│   ├── main_RK.f90               -> Main Fortran script
│   └── sub_RK.f90                -> Fortran module containing subroutines used in main_NB.f90
└── postprocess_example         -> [Folder] containing an example script for postprocessing
    ├── NumMethodComparison.py    -> Python script generating result validation plots among different solvers
    ├── d82_4_N30_35Hz_F1p40_dt1eM5_n30_U2.txt -> Example output from Abaqus simulation
    ├── d82_4_N30_76Hz_F0p50_dt1eM5_n30_U2.txt -> Example output from Abaqus simulation
    ├── d82_4_N30_8Hz_F0p10_dt1eM5_n30_U2.txt  -> Example output from Abaqus simulation
    ├── d82_4_N30_96Hz_F0p60_dt1eM5_n30_U2.txt -> Example output from Abaqus simulation
    ├── d82_4_NB_0.1N_8Hz.h5                   -> Example output from NB solver
    ├── d82_4_NB_0.5N_76Hz.h5                  -> Example output from NB solver
    ├── d82_4_NB_0.6N_96Hz.h5                  -> Example output from NB solver
    ├── d82_4_NB_1.4N_35Hz.h5                  -> Example output from NB solver
    ├── d82_4_RK4_0.1N_8Hz_dt1eM5.h5           -> Example output from RK solver
    ├── d82_4_RK4_0.5N_76Hz_dt1eM5.h5          -> Example output from RK solver
    ├── d82_4_RK4_0.6N_96Hz_dt1eM5.h5          -> Example output from RK solver
    └── d82_4_RK4_1.4N_35Hz_dt1eM5.h5          -> Example output from RK solver
```