DENM and sDENM program manual

Daniel R. Martin*
Department of Physics and Department of Chemistry & Biochemistry, Arizona State University, PO Box 871504, Tempe, Arizona 85287

* daniel.martin@asu.edu

CONTENTS

I. INSTALLATION

The compilation of DENM has been successfully tested with both `ifort` and `gfortran`. There is a `makefile` within the `DENM_PROGRAM` directory accompanying this manual. Depending on the compiler used, modification of the makefile is required. That is, if `ifort` is being used as the compiler the line

```
COMP= gfortran
```

needs to be commented out with a hashtag `#` and the line

```
#COMP= ifort
```

needs to be uncommented by removing the hashtag. Furthermore, the line

```
OPT= -fPIC
```

needs to be commented with a hashtag and the line

```
#OPT= -fPIC -shared-intel -fast
```

needs to be uncommented by removing the hashtag. There is another `ifort` option line

```
#OPT= -fPIC -shared-intel -mcmodel=large -fast -O0 -g -traceback -fpe:0 -check all
```

that can be used to run in a debugging mode that prints out information during runtime if there are any problems that arise which would most likely be caused by modification of the code. It must be noted, however, that including these option causes the performance of the code to diminish substantially. The code has been tested with both the `gfortran` and `ifort` compilers.

Compilation of the code is very straight forward. To compile the code, one first changes into the `DENM_PROGRAM` directory and executes the following on the command line

```
>> make
```

The output for `gfortran` should be

```
gfortran -fPIC  -c modules.f90 -o modules.o
gfortran -fPIC  -c aldeArrays.f90 -o aldeArrays.o
gfortran -fPIC  -c general.f90 -o general.o
gfortran -fPIC  -c psfread.f90 -o psfread.o
gfortran -fPIC  -c pdbread.f90 -o pdbread.o
gfortran -fPIC  -c numRec.f90 -o numRec.o
gfortran -fPIC  -c math.f -o math.o
gfortran -fPIC  -c other.f90 -o other.o
gfortran -fPIC  -c hessian.f90 -o hessian.o
gfortran -fPIC  -c main.f90 -o main.o
gfortran -fPIC  -o denm.e modules.o aldeArrays.o general.o psfread.o pdbread.o numRec.o math.o
                   other.o hessian.o main.o
```

The compilation will produce the executable file `denm.e`.

II. INTRODUCTION

The Dissipative Electro-elastic Network Model (DENM) is used to calculate the statistics and dynamics of protein electrostatic interactions using as a base the elastic network model (ENM) and the protein charge distribution. This manual describes the technical methodology to run DENM calculations corresponding to version 1.0 of the program. The program version described here was used to produce the results in Ref. 1 and 2. In Section III, we describe the overall format of the input file and how it is used to perform various different calculations. Section IV describes how to construct PDB (coordinates) and PSF (charges) files needed to run all the calculations. There the construction of the accompanying PDB and PSF files are performed using VMD and its tcl scripting interface. In section V A, we show the general use of the program using cytochrome B as an example structure and describe most of the output files that are a result of running the program. This section will also cover how to calculate the total displacement response function from the output of the program. In section V B, we cover the fluctuations of the mean-squared displacements and the displacement response functions produced by DENM. In section V C, the electrostatic potential and electric field response due to a probe charge at the iron heme position in cytochrome B are calculated. In section V D, the dielectric response setup and calculation is covered, while in section V E, the charge transfer response setup and calculation is covered. Finally, sections VI A and VI B will cover how to include the solvation of charged residues generally and in the allosteric response functions. The appendix has a complete list of the subroutines with descriptions and the overall program structure.

III. THE INPUT FILE

A. The Input File Description

Here we describe a general method used to create the input file to run the DENM program. There are several calculations one can perform using DENM all involve the creation of an input file and corresponding structure (PDB) and charge (PSF) files. Generally, the calculations are controlled by flags which turn on and off certain aspects of the program and the input file is how to tell the DENM program what to calculate. With each flag (0 or 1), there are corresponding lines in the input file that are either included or not, but the overall structure of the input file is fairly consistent. Each specific calculation may or may not require certain other lines to be included or excluded for a particular given flag. When some flags are turned on (setting to 1), some output may be suppressed even if specified in the input file. This will become more clear throughout the manual. We have color coded the example input file line numbers in this manual to emphasize the need for inclusion of certain other lines within the file (Section III B). It is also important to mention here that due to the way the program handles "adjacent" nodes in the network, the order of the amino acid chain or residues in the PDB/PSF is very important. This means that the way the program identifies the adjacent nodes is due to the sequence within the PDB/PSF files. This will become more clear as we proceed with the example calculations. Here we will discuss the lines in the input file one-by-one.

The first two entries in the input file (1 and 2 below) are the PDB file and the PSF file, respectively. The PDB file contains the coordinates of the atoms that will be used in the program and the PSF file contains the partial charges of the atoms. Both files contain the atom indices, residue numbers, residue names, and various other information. These files must have the same exact order of atoms. Both files are required to run the program and must have entries with target atoms named (typically alpha carbons) CA, or CAP as an option (SECTION). The atoms with the label CA will be treated as the nodes connected by springs.

The next line (3) has the number of residues, n, for which a displacement response function will be produced. These correspond to Fig.3(a) in Ref. 1. That is, $\chi_i(\omega)$ is produced for each residue on the n residues listed below this line (4 and 5). These lines provide a means of calculating many self response functions without having to run calculations for each residue separately. The n following lines (4 and 5) are the residue identification numbers (resid) for each of the n residues referenced directly from the accompanying PDB file.

The next line (6) after the resid's is the cutoff distance measured in Angstroms. For a given atom centered at $\mathbf{r}_0$, all atoms whose position lies outside a sphere of radius $r_{\rm cut}$ away from $\mathbf{r}_0$ will not be connected to the center atom by a spring. Line (7) is the tolerance of the eigenvalues. All eigenvalues with a value less than this number will be considered zero (there should be 6 zeros for any given calculation). The next line (8) is the force constant in units of Å^2 defined as $(\beta C)^{-1}$ where $\beta = 1/(k_B T)$. Line 9 is value of the low-frequency friction coefficient ζ_l in units of nanoseconds followed by the high-frequency friction coefficient (line 10) defined as f_h where $\zeta_h = f_h \zeta_l$. The relative amplitude, a, of the high-frequency term in the two Debye form of the response function

$$\chi_m(\omega) = \frac{a}{i\omega\zeta_h + \lambda_m} + \frac{1-a}{i\omega\zeta_l + \lambda_m} \tag{1}$$

is given on line 11.

Line 12 is related to the force constant of the backbone atoms. The value given on this line allows one to change the strength of the spring attached between adjacent nodes without affecting the springs between non-adjacent nodes. Effectively, it is a coefficient that multiplies the original force constant such that the adjacent nodes have force constant $C_{\rm bb} = f_C C$, where $C_{\rm bb}$ is the derived backbone force constant and f_C is the numerically value in the input file on this line (line 12). The next line (13) is another spring constant added for more flexibility in the program. It allows one to define adjacent nodes to nodes labeled with the name CAP, a different force constant which is the multiplication of the number on this line with the original spring constant, $f_{\rm CAP}C$.

Line 14 in the input file is a flag that tells the program to calculate the response to a probe charge (0-orange), the donor/acceptor response between two states associated with charge transfer due to partial charge differences Δq for a given subset of atoms (1-yellow), and the dielectric response (2-purple). If the value 0 (for probe charge) is given for this flag (14), then the index of the probe atom from the pdb file needs to be provided on the next line (15), followed by the charge on that probe atom (16). If the flag is set to 1 (charge transfer) (14), the following lines (15) and 16) are the number of atoms with Δq charges (15-yellow) and the file that contains the values of the Δq's (16). The format of this file will be considered below in Section V E. If this flag is set to 2 (14), indicating both probe charge and dielectric response calculations, the following lines must be the index of the probe charge (15), the value of the probe charge in units of elementary charge (e) (16), the name of the output file that will contain the charges of the individual residues (17), and the name of the output file (18) that will contain the dielectric response function.

The next two lines are flags that turn off (0) and turn on (1) the calculation with the solvation of charged residues (19) and the allosteric response (20), respectively. If the allosteric flag is set to 1, then the next three lines (21–23) pertain to the allosteric calculation (Section VI B). The first of these 3 lines (21) is the name of the PDB file containing the coordinates of the structure to compare to the original PDB file between the two states giving rise to the allosteric effect. This file must have the same number of atoms as in the PDB file referenced on line 1 in the input file. The next two lines (22 and 23) are the names of the corresponding output files, i.e. the file containing the frequency dependent response function $\delta r_i(\omega)$, Eq. (27) of Ref. (2), (22) and the file containing the frequency dependence of the projection of response along the direction between nodes (23). These lines are to be omitted if the allosteric flag is set to 0.

If the solvation flag is set to 1, the next lines (24–28) pertain to the solvation response calculations. The lines in the input file here are the name of the file containing the values of the Solvent Accessible Surface Area (SASA) (24) (discussed below Section VI A), the value of s in Eq. (13) from Ref. (2) (25), the value of the solvent dielectric constant (26), the value of A in Eq. (13) from Ref. (2) (27), and the cutoff fraction, $f_{\rm sasa}$, for the surface area in the solvation calculation (28). This last parameter is used to determine which residues are involved in the solvation calculation. That is, if the total surface area is given by $\sigma_{\rm SA} = 4\pi s^2$, then residues with SASA's less than $f_{\rm sasa} \times \sigma_{\rm SA}$ are not included in the calculation.

The next three lines (29–35) pertain to the frequency output of the response function. The first of these lines (29) is the number of ω points there will be in the output files. This number is directly proportional to the amount of memory required to run the program, so it should be used sparingly or run in smaller ranges of ω to reduce the memory usage. The two lines (30 and 31) are the initial and final value of ω. These lines are followed by the output file names for the eigenvalues (32), λ_m, the diagonal components of the mean-squared displacements (33), $\left\langle (\delta r_i^\alpha)^2 \right\rangle$, the probe charge electrostatic potential response function (34), the probe charge electric field response function (35), and a list of file names for the response function for individual residues corresponding to $\chi_i(\omega)$ (36 and 37), respectively. Lines 36 and 37 correspond to the residues listed in lines 4 and 5, with the number of residues on line 3. The last five lines (38–42) are two residue numbers (38 and 39) for the calculation of $\chi_{ii}(\omega)$, $\chi_{jj}(\omega)$, and $\chi_{ij}(\omega)$ followed by the three corresponding file names (40–42). These last lines are independent of the lines 3–5 and 36–37 as in the example input file.

B. An Example Input File

```
<PDB FILE>.pdb
<PSF FILE>.psf

2                               #number of individual residues to print
20                              #resids of individual residues to print
54

15.0                            #cutoff
1.0e-5                          #eigencut
1.0                             #gammap - spring constant
30.0                            #zeta - friction coefficient
0.006                           #zetaH
0.35                            #ampl
100.0                           #epsi - controls backbone NN spring strengths
1.0                             #epsi1

0                               #potFlg - 0, 1, or 2
802                             #a0index - probe atom index
1.0                             #q0 - probe atom charge

1                               #potFlg - 0, 1, or 2
101                             #number of dq atoms
<FILE DQ>                       #file with dq charges

2                               #potFlg - 0, 1, or 2
802                             #a0index - probe atom index
1.0                             #q0 - probe atom charge
resQ.dat
chiDielResp.dat

1                               #solvFlg - 0 or 1

1                               #allostFlg - 0 or 1
<REFERENCE PDB FILE>            # for allosteric comparison
DelRVsOmega.dat
drVsOmega.dat

<SASA FILE>
4.4
78.0
3.54
0.16

1000                            #dbn - omega length
0.0001                          #dbs - omega start
10000.0                         #dbf - omega finish
eigenv.dat
fluc2.dat
chiVsOmega.dat
chiEVsOmega.dat

chiVsOmega_20.dat
chiVsOmega_54.dat

20                              #i: CHIii, CHIij
54                              #j: CHIjj, CHIij
chi20.dat
chi54.dat
chi20_54.dat
```

IV. PREPARATION OF THE PSF AND PDB FILES

The first step in any calculation using DENM is to create the corresponding PDB and PSF files. This task will be accomplished using VMD's psfgen plugin and the CHARMM force-field charges corresponding to the standard set of amino acids. The heme is parameterized similarly through the CHARMM force-field. Here we show how to prepare the heme protein cytochrome B. The initial PDB file can be found on the PDB website, PDB entry 256B, and is included with the accompanying files. The PDB file, `256B.pdb`, can be found in the directory

```
DENM_v1.0/examples/cytB562/pdb/
```

Within this same directory there is a tcl script to extract the protein and the heme, `cytB256.extract.tcl`. It can be run by executing the following command within the above directory

```
>> vmd -dispdev text -eofexit < cytB256.extract.tcl
```

Executing this command will generate the file `cytB256.protein_heme.pdb` containing PDB only information directly form the original PDB file for the protein and heme structure (not including charges or mass). To create the PDB and PSF file to run the DENM program, we will need to execute the following VMD command on the tcl file `all_psfgen.cytB256.tcl`

```
>> vmd -dispdev text -eofexit < all_psfgen.cytB256.tcl
```

This produces the PDB and PSF files `cytB256.all.pdb` and `cytB256.all.psf`, respectively, by applying the force-field parameters and partial atomic charges. Once these files are created, one then needs to manually change the name of the iron atom in the heme to CA in both the PDB and PSF files. Explicitly, the line in the PDB file needs to be changed from

```
ATOM   1653  FE  HEMEA 109      -2.615   8.611   0.019  1.00  0.00      H   FE
```

to

```
ATOM   1653  CA  HEMEA 109      -2.615   8.611   0.019  1.00  0.00      H   FE
```

and in the PSF file, from

```
1653 H    109  HEME FE   FE     0.240000       55.8470           0
```

to

```
1653 H    109  HEME CA   FE     0.240000       55.8470           0
```

These modified files are in the directory as `cytB256_CA.pdb` and `cytB256_CA.psf`.

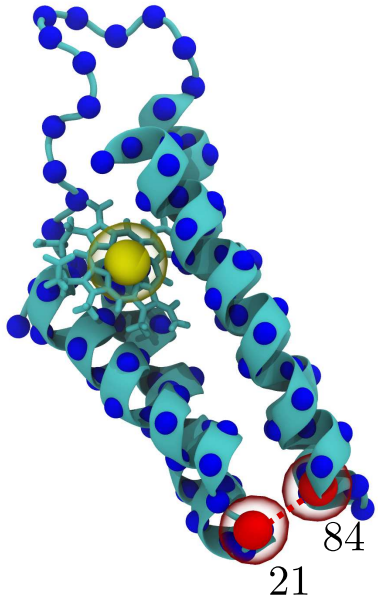

FIG. 1. Cytochrome B (PDB:256B). The dark blue spheres are the C_α atoms which are used as the nodes or beads in the DENM calculation. The red spheres (residues 21 and 84) are the nodes used for the displacement calculations and the yellow sphere is the iron atom of the heme used as the node of the heme and location of the probe charge for the electrostatic potential and electric field calculations.

V. PROGRAM USE - DENM

A. General Use

As an example of how to use the DENM program, we have put together some example calculations using the heme protein Cytochrome B (shown in Fig. (1) and the PDB and PSF files produced in Section IV. The input file for these calculations can be found in the directories `den/examples/cytB256/zeta5` and `DENM_v1.0/examples/cytB256/zeta30`, with name `cytB.general.in`. After changing in to the directory `DENM_v1.0/examples/cytB562/zeta5`, from the command line, DENM can be executed with the following command

```
./denm.e < cytB.general.in >& cytB.general.out &
```

After completion, the following files will be produced

- `eigenv_CA.dat` - The indexed eigenvalues in ascending order.
- `fluc2_CA.dat` - The x, y, z, and total mean squared displacements (columns 3, 4, 5, and 6, respectively) along with a generic index (column 1) and the residue number from the PDB/PSF file (column 2).
- `resQFile.dat` - Contains an index, the `resname` from the PDB/PSF file, the `resid` from the PDB/PSF files, and the total residue charge, respectively.
- `chiVsOmega_21.dat` - The ω-dependent displacement response function for residue 21. This corresponds to the residue number listed in the upper part of the input file specifying the number of residues to print the response function. The format of this file is ω, $\chi_i'(\omega)$, $\chi_i''(\omega)$. This is similar to lines 3 and 4 in the input file provided in Section **??**.
- `chiVsOmega_84.dat` - The ω-dependent displacement response function for residue 84. This corresponds to the residue number listed in the upper part of the input file specifying the number of residues to print the response function. The format of this file is ω, $\chi_i'(\omega)$, $\chi_i''(\omega)$. This is similar to lines 3 and 5 in the input file provided in Section **??**.
- `chiEVsOmega_CA.dat` - The ω-dependent electric field response function for the probe charge with index 1653 (the iron atom of the heme) and with unit charge. The format of this file is ω, $\chi'(\omega)$, $\chi''(\omega)$.
- `chiVsOmega_CA.dat` - The ω-dependent electrostatic potential response function for the probe charge with index 1653 (the iron atom of the heme) and with unit charge. The format of this file is ω, $\chi'(\omega)$, $\chi''(\omega)$.
- `chi21.dat` - The ω-dependent displacement response function for residue 21. This corresponds to the residue number listed above this file in the input file. The format of this file is ω, $\chi_i'(\omega)$, $\chi_i''(\omega)$. This is similar to lines 38–42 in the input file provided in Section **??**.
- `chi84.dat` - The ω-dependent displacement response function for residue 84. This corresponds to the residue number listed above this file in the input file. The format of this file is ω, $\chi_i'(\omega)$, $\chi_i''(\omega)$. This is similar to lines 38–42 in the input file provided in Section **??**.
- `chi21_84.dat` - The cross term in the total response function between residues 21 and 84 such that the total response function is given by

$$\chi_{ij}''^{\mathrm{Tot}}(\omega) = \frac{\chi_i''(\omega) + \chi_j''(\omega) - 2\chi_{ij}''(\omega)}{\chi_i'(0) + \chi_j'(0) - 2\chi_{ij}'(0)} \tag{2}$$

  where $\chi_{ij}''(\omega)$ and $\chi_{ij}'(\omega)$ are that provided within this file and $\chi_i''(\omega)/\chi_j''(\omega)$ and $\chi_i'(\omega)/\chi_j'(\omega)$ are provided in the previous two files. There are various Fortran files, `format.Phi_E.f90` and `format.dr.f90`, included within the example file directories that provide a method for combining the output files from DENM to obtain the imaginary part response the response function, i.e. the loss function.
- `cytB.general.out` - The output file for DENM generated at the command line.

We have also included the directory `zeta30` which performs the same calculation as above accept with the low-frequency friction coefficient given as $\zeta_l = 30$ns.

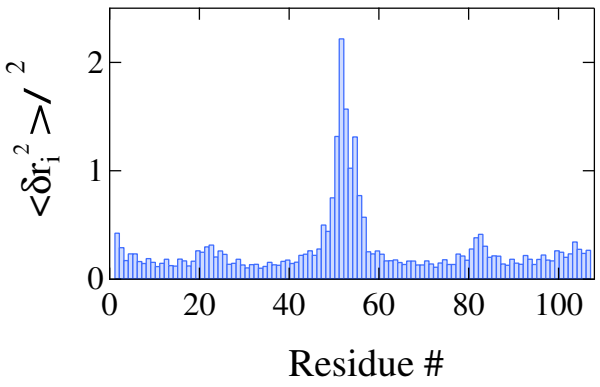

FIG. 2. The total MSD in Å^2 verses residue number of cytochrome B from the output of the DENM program. This result can be produced by plotting column 1 verses column 6 in the file `fluc2_CA.dat`. One can easily see that the most flexible region of the protein is residues between approximately 50 and 60 corresponding to the loop region in Fig. (1).

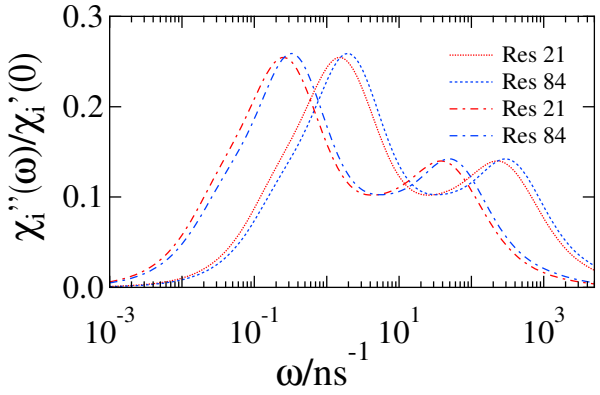

FIG. 3. The results from the files `chiVsOmega_21.dat` and `chiVsOmega_84.dat` for residue 21 (blue) and 84 (red) and for $\zeta_l = 5$ns (dotted) and $\zeta_l = 30$ns (dash-dotted). Generally, one can see the two Debye form of the response function, however, there is an indication of a third component close to the low frequency peak.

B. Residue Displacement Response

The files produced in Section V A pertaining to the displacement statistics and dynamics can be used to create the plots given in this section along with the Fortran programs `format.Phi_E.f90` and `format.dr.f90`. The file `fluc2_CA.dat` contains the mean-squared-displacements (MSD), $\langle \delta x^2 \rangle$, $\langle \delta y^2 \rangle$, $\langle \delta z^2 \rangle$, and $\langle \delta r^2 \rangle$ in Å^2. The plot of column 1 vs. column 6 is shown in Fig. (2) noting that the residue number in this case is just an index and not the residue number from the PDB/PSF file (the only difference being that of the heme which has a PDB residue number of 109 verses the indexed number 107).

Next, we have the files representing the displacement response function, `chiVsOmega_21.dat` and `chiVsOmega_84.dat`, which consequently are the same as `chi21.dat` and `chi84.dat` due to the form of the input file. These can be seen in Fig. (3) for both residue 21 (blue) and 84 (red) and for $\zeta_l = 5$ns (dotted) and $\zeta_l = 30$ns (dash-dotted). The data for these plots can be formed by dividing imaginary part of the response function by the static limit of the real part response function, (column 2). Explicitly, this is given by $\chi_i''(\omega)/\chi_i'(0)$ as indicated by the label of the ordinate (vertical axis).

Finally, for the dynamics of the displacement, we have the cross-correlation term of the displacement response function shown in Fig. (4). The data for this plot can be obtained directly by applying Eq. (2) to the files `chi21.dat`,

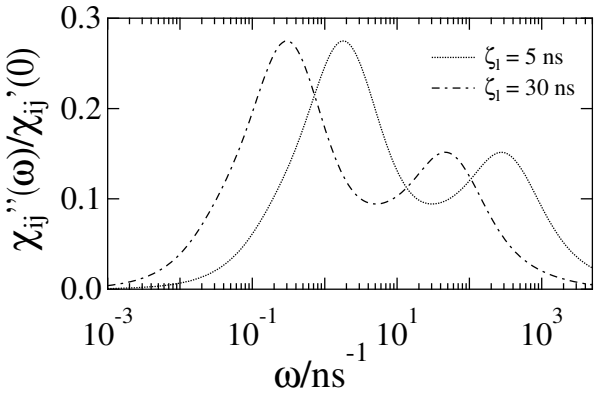

FIG. 4. The results from the files `chi21.dat`, `chi84.dat`, and `chi21_84.dat` residues 21 and 84 and for $\zeta_l = 5$ns (dotted) and $\zeta_l = 30$ns (dash-dotted). The Fortran program `format.dr.f90` was used to put together the mentioned files to produce this plot.

`chi84.dat`, and `chi21_84.dat`. The dotted line represents that obtained for $\zeta_l = 5$ns and the dot-dashed line for $\zeta_l = 30$ns. The Fortran program `format.dr.f90` was used to put together the mentioned files to produce the plot seen in Fig. (4).

C. Electrostatic Field and Potential Response

The electrostatic potential and electric field response are also produced in the above example of Section V A, files `chiVsOmega_CA.dat` and `chiEVsOmega_CA.dat` with the results shown in Figs. (5) and (6), respectively. Here, as was the case for the displacement response, we divide the imaginary part of the response function by the real part evaluate at $\omega = 0$ to obtain the response function shown in the figures. The calculation was done by using the Fortran program `format.Phi_E.f90` given in the directory of the particular DENM run. These two response functions are generated due to lines similar to lines 14–16 in the input file, except here the values pertaining to the iron atom have been added. That is, from the input file used for this calculation, the lines are

```
0                                   #potFlg - 0, 1, or 2
1653                                #a0index - probe atom index
1.0                                 #q0 - probe atom charge
```

where 1653 is the atom index of the iron atom of the heme (labeled CA) for the DENM calculation.

D. Dielectric Response

For the dielectric response, we will continue with the cytochrome B example. For this case, we will change the input file to include the dielectric response, that is, lines 14–18 need to be included as is illustrated in the input file in this manual. The example input file `cytB.general.in` can be found in the directory

`DENM_v1.0/examples/cytB562/diel/noSolv/denm2012`

Within that directory, one will find the input file `cytB.general.in` used to run this calculation. Lines 14–16 have been replaced by lines 14–18 indicating that the dielectric response is being calculated. The pertinent lines are given by

```
2                                   #potFlg - 0, 1, or 2
1653                                #a0index - probe atom index
1.0                                 #q0 - probe atom charge
resQ.dat
chiDielResp.dat
```

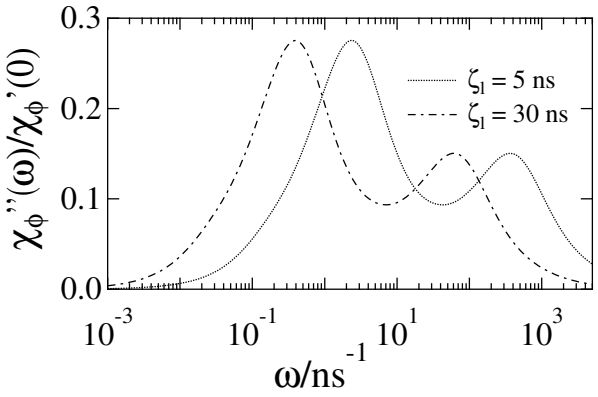

FIG. 5. The electrostatic potential response function due to the probe charge at the heme iron atom of cytochrom B with the probe charge set to unity. The dotted line represents the result for the friction coefficient $\zeta_l = 5$ns and the dash-dotted line that with $\zeta_l = 30$ns as indicated inside the plot. These curves were generated from the file `chiVsOmega_CA.dat` in the subsequent directories for the different values of ζ_l and with the use of the Fortran program `format.Phi_E.f90`.

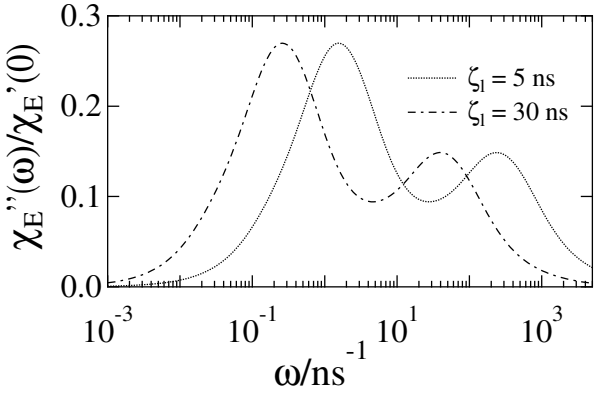

FIG. 6. The electric field response function due to the probe charge at the heme iron atom of cytochrom B with the probe charge set to unity. The dotted line represents the result for the friction coefficient $\zeta_l = 5$ns and the dash-dotted line that with $\zeta_l = 30$ns as indicated inside the plot. These curves were generated from the file `chiEVsOmega_CA.dat` in the subsequent directories for the different values of ζ_l and with the use of the Fortran program `format.Phi_E.f90`.

After running DENM on this input file, the file `chiDielResp.dat` is created which contains the raw data needed to produce the calculated dielectric response. This result can be seen in Fig. (7). The Fortran program `format.Diel.f90` was also used to calculate the loss function. The blue curves in Fig. (7) show the results of running with solvation turned on (sDENM) and with a low-frequency friction coefficient $\zeta_l = 30$ns (see Section VI A and VI B). This change of ζ_l is indicated by the shift of the curves to lower frequency and is not due to the addition of of the solvation terms renormalizing the network as can be seen in Fig. (7) in Ref. 2.

E. Charge Transfer Response

In order to show the use of the DENM program for charge transfer reactions, we have included sample calculations for the Reaction Center protein involved in charge transfer of electrons in photosynthesis. The reaction we have set up here is that from the quinone Q_A to the quinone Q_B. This reaction takes place on microsecond timescales and involves the possibility that the position of Q_B is important. The reaction allows us to show several different setups

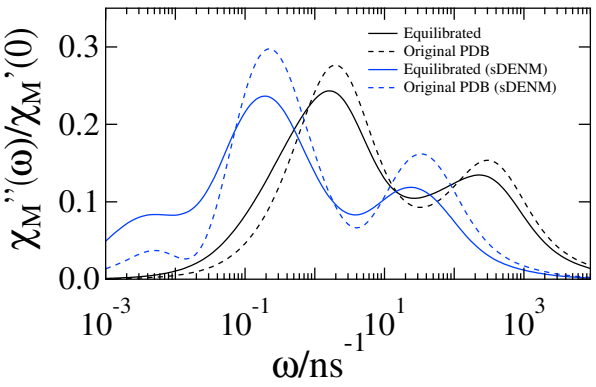

FIG. 7. The dielectric response function of cytochrom B. The solid lines represents the result using a friction coefficient $\zeta_l = 5$ns with a equilibrated structure while the dashed curves represents the calcualtion done with no modifications to the original PDB file as in Sections V A–V C. The curves was generated from the file `chiDielResp.dat` in the subsequent directories after dividing by the real part at $\omega = 0$, $\chi'_M(0)$. The blue curves are the same calculation done with the addition of the solvation of charged residues (sDENM) and with a low-frequency friction coefficient $\zeta_l = 30$ns which is why there is a shift to lower frequencies.

involving some of the more detailed calculations using the DENM program. All of the corresponding files mentioned in this section can be found within the parent directory

```
DENM_v1.0/examples/rc_qaqb/charge_transfer/
```

It must be noted that when the `potFlg` is set to 1, indicating a charge transfer response calculation, certain other output files are suppressed. In particular, the electric field response function, `chiEVsOmega_CA.dat`, is not provided even though it is referenced in the input file, though it is necessary for the program to run properly. Also, the secondary residue displacement response functions are not produced in this calculation. That is, `chi832.dat`, `chiVsOmega_832.dat`, and `chi831_832.dat`, are empty files produced when running the program. This will be corrected in subsequent modification to the code.

Before we perform the calculation of the electrostatic potential response due to the charge transfer reaction, we first need to set up the file for the difference charges between the two states. There are many ways to setup this file, but the important part is to have the correct format. Here, we use VMD's tcl scripting interface in combination with some bash command line techniques to create this file. For illustrative purposes, we have already prepared PDB and PSF files for the Reaction Center for both charge cases, but the methodology to do this is fairly straight forward. That is, there are two configurations from previous simulations with the electron on each of the quinones. Both of these systems have corresponding PDB and PSF files, `rc_qa-minus.extract.pdb`, `rc_qa-minus.extract.psf`, `rc_qb-minus.extract.pdb`, and `rc_qb-minus.extract.psf`. Also, a PSF file that contains the charge differences for the reaction $Q_A^- Q_B \rightarrow Q_A Q_B^-$ has also previously been created, `rc.P+Qa-Qb_To_P+QaQb-.dq.psf`. This last file will be used to create the file we need for the charge transfer reaction response calculation in the DENM program.

The tcl script, `makeDQFile.tcl`, used to prepare a list of donor/acceptor atoms and there charge differences can be found in the directory

```
DENM_v1.0/examples/rc_qaqb/molPdbPsf/
```

This file can be run by executing the command

```
vmd -dispdev text -eofexit < makeDQFile.tcl &> makeDQFile.tcl.out
```

at the command line. It produces an output with the pertinent information tagged with "`INFO)`" which we will use to extract the information needed to create the final charge difference file used by DENM. We can create a list of acceptor atoms by using the `grep` command as follows

```
grep "INFO) " makeDQFile.tcl.out | grep " A " > ACCEPTOR_LIST.dq.txt
```

and executing the command

```
grep "INFO) " makeDQFile.tcl.out | grep " D " > DONOR_LIST.dq.txt
```

creates the list of donor atoms. Finally, to format the file properly, we execute the two commands

```
awk '{printf "%s    %s   %s   %9.6f\n",$2,$3,$4,$5}' ACCEPTOR_LIST.dq.txt > ACCEPTOR_DONOR_LIST.dq.txt
awk '{printf "%s    %s   %s   %9.6f\n",$2,$3,$4,$5}' DONOR_LIST.dq.txt >> ACCEPTOR_DONOR_LIST.dq.txt
```

which creates the file `ACCEPTOR_DONOR_LIST.dq.txt` having the proper format needed to run the DENM program for the charge transfer response calculation. Now, just as a check that we did everything correctly, we can sum the charges in the file with the command

```
awk '{ sum+=$4} END {print sum}' ACCEPTOR_DONOR_LIST.dq.txt
```

which produces the output

```
>> -4.23273e-16
```

which is essentially zero as it should be for a charge transfer between donor and acceptor complexes, $\sum_{\mathrm{D,A}} \Delta q_i = 0$ where the sum is over the donor/acceptor atoms. The `ACCEPTOR_DONOR_LIST.dq.txt` file has the format

```
94302    832   A    0.092348
94303    832   A   -0.039190
94304    832   A   -0.056347
94305    832   A   -0.172488
94306    832   A    0.191062
94307    832   A   -0.447389
.        .     .    .
.        .     .    .
.        .     .    .
```

where the columns are given as the atom index in the PDB/PSF files, the residue id number from the PDB/PSF files, A or D for acceptor atom or donor atom, and the value of Δq_i for that particular atom noting the the acceptor atoms must be listed first within this file.

Now, that we have prepared the donor/acceptor atom file, we can proceed to the actual calculation of the potential response. The first calculation we will perform for this system can be found in the directory

```
DENM_v1.0/examples/rc_qaqb/charge_transfer/qa/run1
```

After labeling the appropriate atoms with the name `CA` within the PDB and PSF files (`molPdbPsf/rc_qa-minus.extract_CA.pdb` and `molPdbPsf/rc_qa-minus.extract_CA.psf`) corresponding the the cofactors and the non-heme iron atom and creating the input file for this calculation (`rc_qa-qb.general.in`), we can execute the DENM program with

```
./denm.e < rc_qa-qb.general.in &> rc_qa-qb.general.out &
```

The input file for this calculation has the corresponding changes on lines 14–16.

The program will take several hours to complete and 9 GB of memory with the ω points set to 200. It is important to note that the amount of memory used in a particular calculation is directly proportional to the number of ω points that will be output. For instance, doing the calculation on this system with 200 ω points will require 9.2 GB of memory, whereas decreasing this number to 100 will only require 4.7 GB of memory. One can break up the calculation into multiple calculation by doing different parts of the range of ω on separate runs. This effectively allows one to "parallelize" the code without have to use true parallelization methodology. It does, however, still require the same amount of memory usage since each individual run will require part of the total memory needed and the sum of this memory will be approximately the total memory needed for a single calculation if not more.

Upon completion, the program will have created output files similar to previous calculations. Here, we are interested in the file `chiVsOmega_CA.dat` which contains the potential response due to the donor/acceptor pair. It must be noted that when performing this calculation, most of the other output is suppressed. That is, the displacement response is only calculated for the first residue listed in the input file and the electric field response is not calculated even though the file `chiEVsOmega_CA.dat` is created and is required in the input file. This will be changed in further versions of the program so that at least the displacement response functions for all listed residues will be calculated.

It also must be noted that since the cofactors are not physically bonded to the protein and due to there placement within the PDB/PSF files, they are treated as connected to each other and with the residue proceeding the first cofactor listed in the PDB/PSF files. We can circumvent this problem by introducing a new spring constant associated with atoms with the name `CAP`. Changing the names of the cofactor center atoms to `CAP` allows the control of the spring

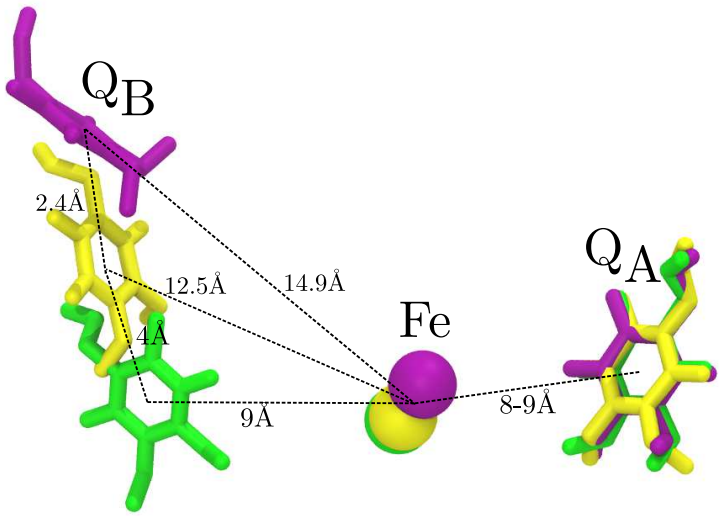

FIG. 8. Diagram showing the cofactors Q_A and Q_B in the Q_A^-/Q_B (purple), Q_A/Q_B^- (yellow), and proximal (green) configurations with respect to the non-heme iron.

constant of those nodes adjacent to the cofactors within the PDB/PSF files. This spring constant is controlled by the `epsil1` described in Section III. One can see the effect of changing the cofactor names to `CAP` and reducing the adjacent spring constant associated with these cofactors by a factor of 0.1 in Figs (9b) and (10). An additional method for circumventing (aside from adopting the code) would be to identify the amino acids to which the cofactors are closest too and move them within the PSF/PDB files to be adjacent to those residues. This method, however, would require renumbering of all the atoms within those files and is beyond the scope of this user manual. Updating the code to include the explicit connectivity matrix would be the best method for dealing with this issue and will be added in new versions.

Without any other modifications except the normal node center atom names (changed to `CA`), the electrostatic potential response due to the charge transfer can be seen in Fig. (9a). In that figure, the black curves represent the calculation done on the structures taken from simulation trajectories we performed for this charge configuration where the cofactor, Q_B, moved in and out of the "pocket". The green curves represent calculations done on structures taken from other simulation trajectories with Q_A and Q_B restrained in the so-called proximal position. The different positions of the cofactors can be seen in Fig. (8).

VI. PROGRAM USE - SDENM

A. Running with Solvation of Charged Residues

Solvation of charged residues can be added to the calculation by setting the `solvFlg` parameter to 1 in the input file on line 19. If this parameter is turned on (by setting it to 1), then lines 24–28 must also be included in the input file noting that if the allosteric flag, `allostFlg`, is set to 0 then lines 21–23 must not be included. The description of the parameters for the solvation flag can be found in Section **??**. As an example calculation, we have calculated the effect on the dielectric response due to charged residues of cytochrome B. Before this DENM calculation can be done, we must first describe how to create the file that contains the solvent accessible surface areas (SASA's). One can create this file in many different ways, but what is important is the format of the file. We have chosen to use VMD to produce this file since it can read PDB/PSF files without any modification and VMD has a plugin that can calculate SASA's for a given structure using an effective van der Waals radius as an input parameter. The pertinent directory where the calculation was performed is

`DENM_v1.0/examples/cytB562/diel/solv`

Within that directory are two sub-directories `denm2012` and `originalPDB`. Within the `originalPDB` directory, there is an example of how to calculate the SASA's for the original PDB file downloaded from the Protein Data Bank,

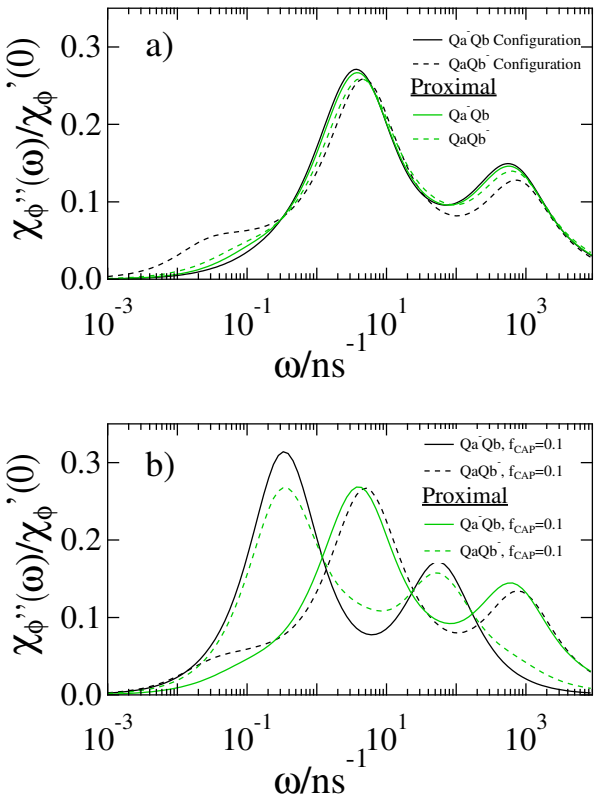

FIG. 9. The result of the calculation of the imaginary part of the electrostatic potential response due to the charge transfer $Q_A^- Q_B \rightarrow Q_A Q_B^-$ in the reaction center. Figure (a) shows the result of no modification of the spring constant between the cofactors. Figure (b) shows the results with the spring constants near the cofactors modified by $f_{CAP} = 0.1$. The black curves represent the results with structures taken directly from simulation data. The green curves represent results taken from structures with Q_A and Q_B in the so-called proximal positions. The charge states are indicated within the plots.

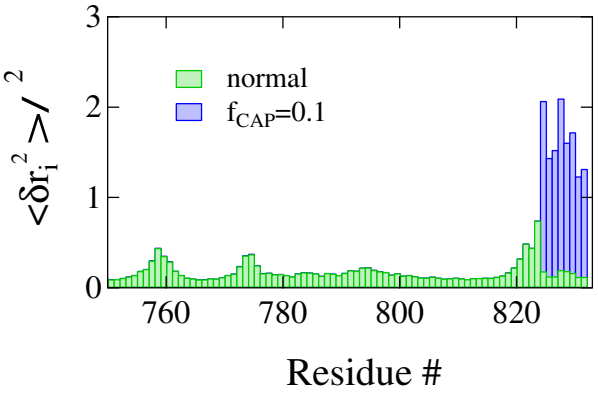

FIG. 10. The fluctuations of the residues from resid 750 to resid 832. The last 9 residues in the plot are the 9 cofactors (including the non-iron heme) within the reaction center. The blue bars represent those results with the $f_{CAP} = 0.1$ modification; That is, with the spring constant for residues near the cofactors reduced by a factor of 0.1.

noting that the hydrogen atoms and the connectivity was applied through VMD's psfgen plugin and the CHARMM forcefield. Using this program to place the hydrogen atoms can create issues when introducing solvation to this model. This direct, bulk approach guesses where the hydrogens belong (from basic chemistry) and can ignore the overall charges of the atoms, thus creating structures that are not minimized in any sense. By attempting to do the solvated DENM calculation without any minimization or equilibration can create instability throughout the network and result in very strange results, such as negative MSD's or a set of eigenvalues that do not specifically contain 6 zero values, or in some cases negative eigenvalues. Therefore, care must be taken when introducing solvation and calculating the SASA's for a given structure. We proceed by showing how to calculate the SASA's for the original PDB file created using VMD, but do not use this file in the calculations. Instead we use a SASA file created from an equilibrated system, thus preventing the spurious results one would obtain by blindly using the structure created by psfgen.

Within the `originalPDB/molPdbPsf/` directory, one will find the file `sasa.tcl` that is generally used to create the formatted file used by DENM for the calculations to be done with the solvation of charged residues. To run this calculation, one can run from the command line

```
vmd -dispdev text -eofexit < sasa.tcl &> sasa.tcl.out
```

This will produce the file `sasa.out.cytB256.dat` which has the format

```
ALA          1          0     1.0000    86.8478
ASP          2          1    -1.0000    86.5070
LEU          3          2     0.0000    10.8581
GLU          4          3    -1.0000    92.5767
ASP          5          4    -1.0000    75.1154
.            .          .     .         .
.            .          .     .         .
.            .          .     .         .
```

Column by column, these entries represent the residue name, residue id number, an indexed residue number starting from 0 and ending with the total number of residues, the charge of that residue summed from the PSF file, and the SASA given as the output from VMD's SASA plugin, respectively. This file is the in the format needed by DENM to run the calculations with the solvation of charged residues.

Now that we have provided a method for creating the SASA file, we will turn to the actual calculation where we use the file `sasa.cytB.txt` that has the same entries as above, but instead calculates the SASA from a pre-equilibrated structure. This SASA file can be used with the original PDB structure to produce the dielectric response as was done in Section V D except with the inclusion of the appropriate changes to the input file. We have performed the calculation of the dielectric response with the solvation of charged residues for both the original PDB structure and an equilibrated PDB structure from a previous simulation. The results of both of these calculations can be seen as the blue curves in Fig. (7).

B. Allosteric Response

As an example of of the application top allosteric response, we will show how to setup the calculation of the scalar displacement response in Eqs. (26) and (27) from Ref. 2. The structures that will be used for this calculation are the unphosphorylated (denoted NtrC) and phosphorylated (denoted P-NtrC) for the single-domain signaling protein NtrC. The phosphordonor carbamoylphosphate (PHD) binds to this protein in large access solutions at the active site Asp54 and shows how a local change in the addition of a charged ion can propagate through the network of residues to create nonlocal perturbations. We will compare the results using both the normal DENM and solvated DENM programs and see that having the solvation of charged residues included indeed creates much larger perturbations in the frequency dependent displacement response at nonlocal locations in this protein.

From a technical standpoint, this calculation will involve having a file that includes the SASA's for the two protein structures (with adhoc modifications to stabilize the network) as well as the addition of the allosteric input parameters (lines 20–23) in the input file. All the following calculations can be found in the parent directory

```
DENM_v1.0/examples/NtrC/
```

If we load original PDB files, PDB entries 1DC7 (NtrC unbound) and 1DC8 (P-NtrC bound), found in the directory

```
DENM_v1.0/examples/NtrC/build/origPDB/
```

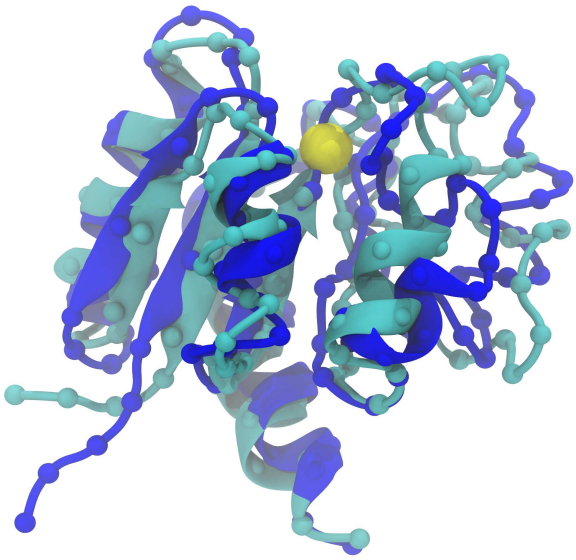

FIG. 11. A cartoon of the aligned structures for NtrC and P-NtrC with the binding site shown in yellow.

into VMD, it is fairly obvious that these files are not aligned. In order to perform the calculation of the scalar displacement response, we must first extract the protein and align the two structures. Before we move to the alignment of the structures we must first extract the protein from the PDB files. This can be done by using the tcl scripts `NtrC.extract.tcl` and `P-NtrC.extract.tcl` and running the following commands from the command line

```
vmd -dispdev text -eofexit < NtrC.extract.tcl
vmd -dispdev text -eofexit < P-NtrC.extract.tcl
```

These commands will produce the files `NtrC.pdb` and `P-NtrC.pdb` with the pertinent structures with Asp54 in the NtrC structure and the PHD structure in the P-NtrC structure.

Once the protein have been extracted from the original PDB files, we must do an alignment of the two structures. There are several programs that can do this type of alignment which involves calculation of the principal axes of the inertia tensor and rotations/translations to move the structures into alignment. Here we move the P-NtrC structure to coincide with the NtrC structure. The method described here uses VMD's `measure fit` and `move` functions. First, load both structures into VMD (1DC7 first followed by 1DC8) and then execute the following commands in the tcl/tk console:

```
set sel0 [atomselect 0 "protein and name CA"]
set sel1 [atomselect 1 "protein and name CA"]
set M [measure fit $sel1 $sel0]
set all1 [atomselect 1 all]
$all1 move $M
$all1 writepdb P-NtrC-align.pdb
```

These commands align the second structure loaded to the first structure and then write a PDB file with the aligned second structure, `P-NtrC-align.pdb`. The aligned structures can be seen in Fig. (11). For the commands to work properly, VMD must be started fresh. In other words, since references are made to "0" and "1" in the `atomselect` statements above, they must coincide with the correct structures loaded in VMD as can be seen in the main graphics window when VMD is opened.

After running the DENM program on the input files in the directories, a number of files are produced including the files already described. For instance, the MSD's from the calculation are given in the file `fluc2.dat`. A plot of column 2 verses column 6 for the three DENM runs calculated (unbound-DENM, boun-DENM, and bound-sDENM) can be seen in Fig. (12). This plot shows that including solvation has a large affect on the corresponding residue fluctuations.

The file `DelRVsOmega.dat` is also produced. This file contains the function $\delta r_i(\omega)$ as describe in the text around Eq. (26) and (27) in Ref. (2) for ALL residues and therefore must be parsed properly to obtain files for each individual

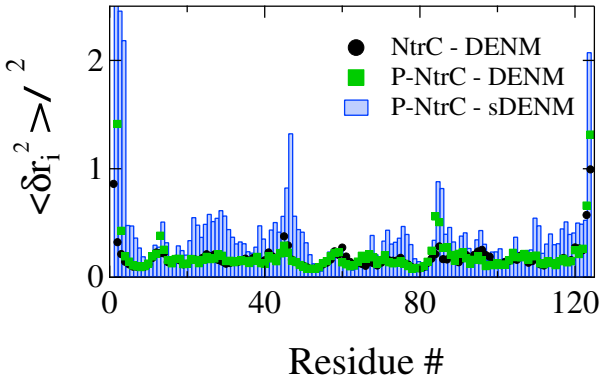

FIG. 12. A comparison of the residue fluctuations between the NtrC protein in both the unbound and bound formed calculated with DENM (black and green, respectively) and the fluctuations calculated in the bound state using the solvated DENM.

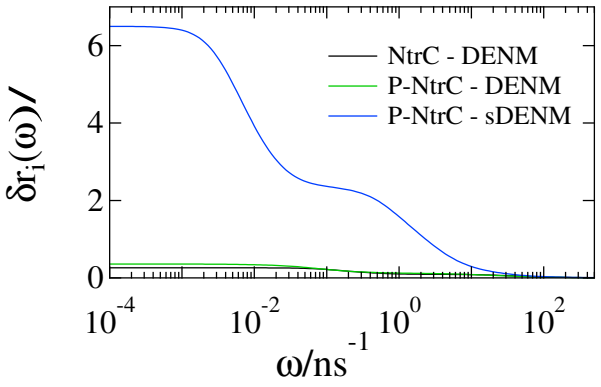

FIG. 13. Frequency-dependent allosteric displacement for residue i=124 (Glu) of NtrC in the unphosphorylated (NtrC) and phosphorylated (P-NtrC) states. The results of calculations within DENM and sDENM are compared as shown in the plot.

residue. For instance, to obtain the response of residue 124, we could run the following commands from the command line

```
grep "RES:      124" DelRVsOmega.dat | awk '{printf "%f   %f\n",$3,$6}' > DelRVsOmega.124.dat
```

This command will produce the file for just the residue 124. For all three DENM runs mentioned, we have produced the plot of the frequency dependence of the scalar displacement. This can be seen in Fig. (**??**). One can see in this plot that having the solvation terms included produces a considerable effect on the network response of the scalar displacement.

Appendix A: Program Structure

Here we provide a list of files and subroutines as well as individual subroutine calls.

1. `main.f90` : Main program - `denmMain`

   (a) `CALL eofile(dqInt,dqFile,eofInt)` - Gets the number of lines in the file containing the Δq for the donor/acceptor calculation. Performed to check that the number of lines in the Δq file is the same as that specified in the input file (15). This subroutine is in the file `psfread.f90`.

   (b) `CALL readDQ(dqFile,dqInt)` - Populates the arrays `indAD`, `resAD`, `ADtxt`, and `dq` related to the donor/acceptor calculation. These correspond to the values in the `<FILE DQ>` listed in the input file (16). This subroutine is in the file `psfread.f90`.

   (c) `CALL setup` - Gets the total number of atoms from the PSF file, `natom`, allocates the main arrays (`resid`, `residc`, `resname`, `aindex`, `atomname`, `atomtype`, `atom`, `charge`, `mass`, `pdbresid`, `pdbresname`, `pdbaindex`, `pdbatomname`, `coor`, `beta`, `segname`, `x`, `y`, and `z`), reads the PSF files, allocates the C_α arrays (the same type of arrays allocated for all the atoms listed previously), and finally reads the PDB file. This subroutine is in the file `general.f90`.

   (d) `CALL popResQ(nres,natom)` - Populates the file `resQ.dat` which contains an index starting from 1 and going to the number of residues, the `resname` and `resid` from the PDB file, and total charge of the residue summed from the PSF file. This subroutine is only called when `potFlg` is 0 (14–16) or 2 (14–18) and can be found in the file `other.f90`.

   (e) `CALL findChargeSasa(nres,nqn0)` - Determines the number of residues with zero and nonzero charges as well as determines the number of residues that meet the surface area criteria corresponding to lines 25 and 28 of the input file above. This subroutine can be found in the file `general.f90`.

   (f) `CALL popSasa(nres,nqn0)` - Populates the array `resYN`. This integer array contains 0 for non-charged residues that meet the surface area criteria, 1 for charged residues that meet the surface area criteria, 2 for charged residues that do not meet the surface area criteria. This subroutine is only called if the `solvFlg` is 1 (19). It can be found in the file `other.f90`.

   (g) `CALL alHessian(res3,natom,dbn)` - Allocates the arrays `hess`, `invhess`, `v`, `w`, `indx`, `flucx2`, `flucy2`, `flucz2`, `hbeta`, `flcAllx`, `flcAlly`, `flcAllz`, `chi`, `chidbr`, `chidbi`, and `chiEdbr`.

   (h) `CALL alElectro(nres)` - Allocates the arrays `field`, `force`, and `qT`.

   (i) `CALL alDeltaR(nres)` - Allocates the arrays `allostDeltaRR`, `allostDeltaRI`, `allostDeltaR`, `allostDeltaRR1`, `allostDeltaRI1`, and `allostDeltaR1`. This subroutine is only called if `allostFlg` is 1 (20) and can be found in the file `aldeArrays.f90`.

   (j) `CALL alDr(nres)` - Allocates the array `DeltaRCAHat`. This subroutine is only called if `allostFlg` is 1 (20) and can be found in the file `aldeArrays.f90`.

   (k) `CALL calcDeltaRCA(natom,nres,pdbUnbFile,pdbUnbInt,1)` - Calculates the unit vectors along the x, y, and z difference distances between the corresponding structures for the allosteric calculations as well as the magnitude of the displacement. As a result the array `DeltaRCAHat` is populated. It also produces a file, `DeltR.Allosteric.dat`, containing the `resname`, `resid`, the x, y, and z displacements, and the magnitude of the displacement. This subroutine is only called if `allostFlg` is 1 (20) and can be found in the file `other.f90`.

   (l) `CALL constructHess(resnum)` - Constructs the normal Hessian corresponding to Eq. (1) in Ref. 1 and Eq. (2) in Ref. 2. This subroutine can be found in `hessian.f90`.

   (m) `CALL calcKappa(nres,nqn0)` - Calculates the second rank tensor representing the dipolar response of the solvent dipole, $\kappa_{ij}^{\alpha\beta}$, corresponding to Eq. (13) in Ref. 2. This subroutine is only called if `solvFlg` is 1 (19) and can be found in the file `other.f90`.

   (n) `CALL constructHessSolv(resnum)` - Constructs the solvated Hessian corresponding to Eq. (5) and (6) in Ref. 2. This subroutine is only called if `solvFlg` is 1 (19) and can be found in the file `hessian.f90`.

   (o) `CALL SVDCMP(hess,res3,res3,res3,res3,w,v)` - The numerical recipes subroutine to singular value decomposition. This subroutine can be found in the file `numRec.f90`.

   (p) `CALL indexx(res3,w,indx)` - The numerical recipes subroutine to put the eigenvalues in ascending order. This subroutine can be found in the file `numRec.f90`.

(q) `CALL calcZeros(res3,numzero)` - Determines the number of zero eigenvalues. All values found less than the number on line 7 of the input file are considered zero. This subroutine can be found in the file `other.f90`.

(r) `CALL calcInvConn(res3)` - Calculates the inverse of the connectivity matrix. This subroutine can be found in the file `other.f90`.

(s) `CALL calcFlucts(nres,flcFile,flcInt)` - Calculates the mean-squared-fluctuations, $\langle(\delta r_i)^2\rangle$, defined by the diagonal components of Eq. (2) in Ref. 1. This subroutine can be found in the file `other.f90`.

(t) `CALL fieldForce(natom,nres,q0,a0index)` - Calculates the electric field and the force on atom with index `a0index` and charge `q0`, given on lines 15 and 16 of the input file above. It also populates the dipolar tensor `qT`. This subroutine is only called if `potFlg` is either 0 or 2 (14 or 14, respectively). This subroutine can be found in the file `other.f90`.

(u) `CALL allostericField(nres,q0,a0index)` - The same as `fieldForce` above, except for the allosteric calculations for which the `allostFlg` is 1 (20). This subroutine can be found in the file `other.f90`.

(v) `CALL fieldForceDE(natom,nres)` - The same as `fieldForce` above, except for the donor/acceptor calculations for which the `potFlg` is 1 (14). Here the dipolar tensor is not populated. This subroutine can be found in the file `other.f90`.

(w) `CALL calcChi(res3,dbn,dbs,dbf,cVoFile,cVoInt,cEoFile,cEoInt)` - The main engine for the dynamics calculation. All response functions are calculated in this subroutine. This subroutine can be found in the file `other.f90`. (COME BACK TO)

(x) `CALL deHessian` - Deallocation of the arrays allocated by `alHessian`. This subroutine can be found in the file `aldeArrays.f90`.

(y) `CALL deElectro` - Deallocation of the arrays allocated by `alElectro`. This subroutine can be found in the file `aldeArrays.f90`.

(z) `CALL deDeltaR` - Deallocation of the arrays allocated by `alDeltaR`. This subroutine can be found in the file `aldeArrays.f90`.

(aa) `CALL deDr` - Deallocation of the arrays allocated by `alDr`. This subroutine can be found in the file `aldeArrays.f90`.

2. `general.f90`

(a) `SUBROUTINE findChargeSasa(nr,nq)` - Subroutine to determine the number of residues with zero and nonzero charges. It also determines the number of residues that meet the surface area criteria corresponding to lines 25 and 28 in the above input file.

(b) `SUBROUTINE setup` - Subroutine that gets the total number of atoms from the PSF file, `natom`, allocates the main arrays (`resid`, `residc`, `resname`, `aindex`, `atomname`, `atomtype`, `atom`, `charge`, `mass`, `pdbresid`, `pdbresname`, `pdbaindex`, `pdbatomname`, `coor`, `beta`, `segname`, `x`, `y`, and `z`), reads the PSF files, allocates the C_α arrays (the same type of arrays allocated for all the atoms listed previously), and finally reads the PDB file.

i. `CALL getnum(psffile,psfInt,psfStart,natompsf)` - A call to the subroutine to return the number of atoms in the PSF file `natompsf` and set it to the global variable `natom`. It also returns `psfStart`, the line in the PSF file which the atom list starts indicated by the line tagged with `!NATOM`. This subroutine can be found in the file `psfread.f90`.

ii. `CALL alStructArrays` - Allocation of the arrays `resid`, `residc`, `resname`, `aindex`, `atomname`, `atomtype`, `atom`, `charge`, `mass`, `pdbresid`, `pdbresname`, `pdbaindex`, `pdbatomname`, `coor`, `beta`, `segname`, `x`, `y`, and `z`. This subroutine can be found in the file `aldeArrays.f90`.

iii. `CALL psfread(psffile,psfInt,alphaInt,psfStart,natom)` - Call to populate the arrays `aindex`, `residc`, `resname`, `atomname`, `atomtype`, `charge`, `mass`, and `resid` from the PSF FILE. If `alphaInt` is set to 0, then the C_α arrays are not populated. This subroutine can be found in the file `psfread.f90`.

iv. `CALL alCalphas` - Allocation of the C_α arrays. The number of C_α's is found from the previous call to `psfread` above. This subroutine can be found in the file `aldeArrays.f90`.

v. `CALL psfread(psffile,psfInt,alphaInt,psfStart,natom)` - Call to populate the arrays `aindex`, `residc`, `resname`, `atomname`, `atomtype`, `charge`, `mass`, and `resid` from the PSF FILE. If `alphaInt` is set to 1, then the C_α arrays are populated. The first call to this subroutine above is to get the number of C_α's so that the C_α arrays can be allocated. This subroutine can be found in the file `psfread.f90`.

vi. `CALL pdbread(pdbfile,pdbInt,alphaInt,natom)` - Populates the arrays `pdbaindex`, `pdbatomname`, `pdbresname`, `pdbresid`, `x`, `y`, `z`, `coor`, `beta`, `segname`, and `atom`, and for the C_α arrays `xca`, `yca`, `zca`, `betaca`, `segnameca`, and `pdbaindexca`. These arrays correspond to lines in the PDB file. This subroutine can be found in the file `pdbread.f90`.

(c) `SUBROUTINE exitProg` - Calls `deStructArrays` and `deCalphas` which deallocate the arrays allocated by `alStructArrays` and `deCalphas`. This subroutine can be found in the file `general.f90`.

3. `aldeArrays.f90` - Subroutines for all allocations and deallocations of all the arrays used and shared.

(a) `SUBROUTINE alStructArrays`

(b) `SUBROUTINE deStructArrays`

(c) `SUBROUTINE alCalphas`

(d) `SUBROUTINE deCalphas`

(e) `SUBROUTINE alHessian(num,na,dbn)`

(f) `SUBROUTINE deHessian`

(g) `SUBROUTINE alElectro(nr)`

(h) `SUBROUTINE deElectro`

(i) `SUBROUTINE alDeltaR(num)`

(j) `SUBROUTINE deDeltaR`

(k) `SUBROUTINE alStructArraysDr(na)`

(l) `SUBROUTINE deStructArraysDr`

(m) `SUBROUTINE alCalphasDr(nc)`

(n) `SUBROUTINE deCalphasDr`

(o) `SUBROUTINE alDr(n)`

(p) `SUBROUTINE deDr`

4. `hessian.f90`

(a) `SUBROUTINE constructHess(resnum)` - Constructs the Hessian according to Eq. (1) in Ref. 1 and Eq. (2) in Ref. 2.

(b) `SUBROUTINE constructHessSolv(resnum)` - Constructs the solvated Hessian corresponding to Eq. (5) and (6) in Ref. 2. This subroutine is only called if `solvFlg` is 1 (19).

5. `math.f`

(a) `SUBROUTINE CERROR(Z,CER)` - Numerical computation of the error function of a complex argument, $erf(z)$ where z is a complex number.

6. `modules.f90` - Contains the modules used throughout the program, and the associated global variables and arrays. Here we provide lists of variables and arrays associated with each module.

(a) `MODULE files`

```
CHARACTER(LEN=60) :: psfFile, pdbFile
INTEGER, PARAMETER :: psfInt = 100
INTEGER, PARAMETER :: pdbInt = 101
INTEGER :: psfStart, natompsf, natompdb
```

(b) `MODULE rresidue`

```
INTEGER :: rres
INTEGER, ALLOCATABLE, DIMENSION(:) :: rresi
CHARACTER(LEN=60), ALLOCATABLE, DIMENSION(:) :: rrVoFile
INTEGER, ALLOCATABLE, DIMENSION(:) :: rrVoInt
INTEGER :: res1, res2
CHARACTER(LEN=60) :: chiiiFile, chijjFile, chiijFile
```

```
INTEGER, PARAMETER :: chiiiInt = 300
INTEGER, PARAMETER :: chijjInt = 301
INTEGER, PARAMETER :: chiijInt = 302
```

(c) MODULE dielResp

```
REAL, ALLOCATABLE, DIMENSION(:) :: resQ
CHARACTER(LEN=60) :: chiDielFile
CHARACTER(LEN=60) :: resQFile
INTEGER, PARAMETER :: chiDielInt = 400
INTEGER, PARAMETER :: resQInt    = 401
```

(d) MODULE atoms

```
INTEGER :: natom, ncalpha, nresidue, residProbe
REAL :: qprobe
INTEGER, ALLOCATABLE, DIMENSION(:) :: aindex, resid
CHARACTER(LEN=20), ALLOCATABLE, DIMENSION(:) :: residc, resname
CHARACTER(LEN=20), ALLOCATABLE, DIMENSION(:) :: atomname, atomtype
CHARACTER(LEN=2),  ALLOCATABLE, DIMENSION(:) :: atom
REAL, ALLOCATABLE, DIMENSION(:) :: charge, mass
INTEGER, ALLOCATABLE, DIMENSION(:) :: pdbaindex, pdbresid
CHARACTER(LEN=20), ALLOCATABLE, DIMENSION(:) :: pdbresname
CHARACTER(LEN=20), ALLOCATABLE, DIMENSION(:) :: pdbatomname
CHARACTER(LEN=20), ALLOCATABLE, DIMENSION(:) :: segname
REAL, ALLOCATABLE, DIMENSION(:) :: coor, beta
REAL, ALLOCATABLE, DIMENSION(:) :: x, y, z
INTEGER, ALLOCATABLE, DIMENSION(:) :: aindexca, residca
CHARACTER(LEN=20), ALLOCATABLE, DIMENSION(:) :: residcca, resnameca
CHARACTER(LEN=20), ALLOCATABLE, DIMENSION(:) :: atomnameca, atomtypeca
REAL, ALLOCATABLE, DIMENSION(:) :: chargeca, massca
CHARACTER(LEN=20), ALLOCATABLE, DIMENSION(:) :: segnameca
REAL, ALLOCATABLE, DIMENSION(:) :: coorca, betaca
REAL, ALLOCATABLE, DIMENSION(:) :: xca, yca, zca
INTEGER, ALLOCATABLE, DIMENSION(:) :: pdbaindexca
```

(e) MODULE atomsDr

```
INTEGER :: natomDr, ncalphaDr, nresidueDr
INTEGER, ALLOCATABLE, DIMENSION(:) :: pdbaindexDr, pdbresidDr
CHARACTER(LEN=20), ALLOCATABLE, DIMENSION(:) :: pdbresidcDr, pdbresnameDr
CHARACTER(LEN=20), ALLOCATABLE, DIMENSION(:) :: pdbatomnameDr
CHARACTER(LEN=20), ALLOCATABLE, DIMENSION(:) :: segnameDr
REAL, ALLOCATABLE, DIMENSION(:) :: coorDr, betaDr
REAL, ALLOCATABLE, DIMENSION(:) :: xDr, yDr, zDr
INTEGER, ALLOCATABLE, DIMENSION(:) :: aindexcaDr, residcaDr
CHARACTER(LEN=20), ALLOCATABLE, DIMENSION(:) :: residccaDr, resnamecaDr
CHARACTER(LEN=20), ALLOCATABLE, DIMENSION(:) :: atomnamecaDr, atomtypecaDr
CHARACTER(LEN=20), ALLOCATABLE, DIMENSION(:) :: segnamecaDr
REAL, ALLOCATABLE, DIMENSION(:) :: coorcaDr, betacaDr
REAL, ALLOCATABLE, DIMENSION(:) :: xcaDr, ycaDr, zcaDr
INTEGER, ALLOCATABLE, DIMENSION(:) :: pdbaindexcaDr
REAL, ALLOCATABLE, DIMENSION(:,:) :: DeltaRCAHat
```

(f) MODULE hessian

```
REAL :: gammap, zeta, epsi, epsi1, cutoff, cutoffsq, eigencut
REAL :: zHcoeff, ampl
```

```
REAL, ALLOCATABLE, DIMENSION(:,:) :: hess, invhess, v
REAL, ALLOCATABLE, DIMENSION(:) :: w
REAL, ALLOCATABLE, DIMENSION(:) :: flucx2, flucy2, flucz2, hbeta
REAL, ALLOCATABLE, DIMENSION(:) :: flcAllx, flcAlly, flcAllz
INTEGER, ALLOCATABLE, DIMENSION(:) :: indx
COMPLEX, ALLOCATABLE, DIMENSION(:,:,:,:,:) :: chi
REAL, ALLOCATABLE, DIMENSION(:) :: chidbr, chidbi, chiEdbr, chiEdbi
```

(g) `MODULE electro`

```
REAL, ALLOCATABLE, DIMENSION(:,:) :: field
REAL, ALLOCATABLE, DIMENSION(:,:) :: force
REAL, ALLOCATABLE, DIMENSION(:,:,:) :: qT
```

(h) `MODULE deltaEmod`

```
INTEGER :: dqInt, potFlg, nADatoms
INTEGER, ALLOCATABLE, DIMENSION(:) :: indAD, resAD
REAL, ALLOCATABLE, DIMENSION(:) :: dq
CHARACTER(LEN=2), ALLOCATABLE, DIMENSION(:) :: ADtxt
CHARACTER(LEN=60) :: dqFile
```

(i) `MODULE sasaMod`

```
INTEGER :: solvFlg, nqn0, sasaInt, allostFlg, allostInt, allostInt1, pdbUnbInt
INTEGER, ALLOCATABLE, DIMENSION(:) :: sasaResid, sasaRes, resYN
REAL, ALLOCATABLE, DIMENSION(:) :: sasaQ, sasaA
REAL, ALLOCATABLE, DIMENSION(:,:,:,:) :: qqkappa
REAL, ALLOCATABLE, DIMENSION(:) :: allostDeltaRR, allostDeltaRI, allostDeltaR
REAL, ALLOCATABLE, DIMENSION(:) :: allostDeltaRR1, allostDeltaRI1, allostDeltaR1
CHARACTER(LEN=4), ALLOCATABLE, DIMENSION(:) :: sasaResnm
CHARACTER(LEN=60) :: sasaFile, allostFile, allostFile1, pdbUnbFile
REAL :: s, epsS, A, areaFrac
REAL :: totArea, areaCut
```

(j) `MODULE resij`

```
INTEGER :: dRresi, dRresj
```

(k) `MODULE parms`

```
REAL, PARAMETER :: pi=3.14159265358979
```

7. `numRec.f90`

   (a) `SUBROUTINE SVDCMP(a,m,n,mp,np,w,v)` - Subroutine to calculate the singular valued decomposition of a matrix. This subroutine was taken directly from Numerical Recipes.

   (b) `FUNCTION pythag(a,b)` - An accompanying function required for `SVDCMP` from Numerical Recipes.

   (c) `SUBROUTINE indexx(n,arr,indx)` - Returns the indices (`indx`) of an array such that the values are in ascending order. In this case, it is the list of indices of the ascending eigenvalues.

8. `other.f90`

   (a) `SUBROUTINE calcZeros(res3,numzero)` - Calculates the number of zero valued eigenvalues corresponding to the cutoff given on line 7 of the above input file.

   (b) `SUBROUTINE calcInvConn(res3)` - Calculates the inverse of the connectivity matrix.

   (c) `SUBROUTINE calcFlucts(n,flcFile,flcInt)` - Calculates the fluctuations $\left\langle (\delta r_i)^2 \right\rangle$ and writes them to the file `flcFile` with the corresponding unit integer `flcInt`.

(d) `SUBROUTINE fieldForce(na,nr,q0,i)` - Calculates the electric field and the force on atom with index `i` and charge `q0`, given on lines 15 and 16 of the input file above. It also populates the dipolar tensor `qT`. This subroutine is only called if `potFlg` is either 0 or 2 (14 or 14, respectively).

(e) `SUBROUTINE fieldForceDE(na,nr)` - The same as `fieldForce` above, except for the donor/acceptor calculations for which the `potFlg` is 1 (14). Here the dipolar tensor is not populated.

(f) `SUBROUTINE popResQ(nr,na)` - Populates the file `resQ.dat` which contains an index starting from 1 and going to the number of residues, the `resname` and `resid` from the PDB file, and total charge of the residue summed from the PSF file. This subroutine is only called when `potFlg` is 0 (14–16) or 2 (14–18).

(g) `SUBROUTINE popSasa(nr,nq)` - Populates the array `resYN`. This integer array contains 0 for non-charged residues that meet the surface area criteria, 1 for charged residues that meet the surface area criteria, 2 for charged residues that do not meet the surface area criteria. This subroutine is only called if the `solvFlg` is 1 (19).

(h) `SUBROUTINE calcKappa(nr,nq)` - Calculates the second rank tensor representing the dipolar response of the solvent dipole, $\kappa_{ij}^{\alpha\beta}$, corresponding to Eq. (13) in Ref. 2. This subroutine is only called if `solvFlg` is 1 (19).

(i) `SUBROUTINE calcDeltaRCA(na,nc,filenm,fileInt,alphaInt)` - Calculates the unit vectors along the x, y, and z difference distances between the corresponding structures for the allosteric calculations as well as the magnitude of the displacement. As a result the array `DeltaRCAHat` is populated. It also produces a file, `DeltR.Allosteric.dat`, containing the `resname`, `resid`, the x, y, and z displacements, and the magnitude of the displacement. This subroutine is only called if `allostFlg` is 1 (20).

(j) `SUBROUTINE allostericField(nr,q0,i)` - The same as `fieldForce` above, except for the allosteric calculations for which the `allostFlg` is 1 (20).

(k) `SUBROUTINE allostericFieldCA(nr,q0,i)` - The same as `allostericField` except only on the atoms labeled with a `CA` in the PDB and PSF files. This subroutine is currently not implemented anywhere in the code.

(l) `calcChi(res3,dbnum,dbstart,dbfinal,cVoFile,cVoInt,cEoFile,cEoInt)` - The main engine for the dynamics calculation. All response functions are calculated in this subroutine. (COME BACK TO)

9. `pdbread.f90`

(a) `SUBROUTINE pdbread(filenm,fileInt,alphaInt,num)` - Reads the PDB file and populates the arrays `pdbaindex`, `pdbatomname`, `pdbresname`, `pdbresid`, `x`, `y`, `z`, `coor`, `beta`, `segname`, and `atom`, and for the C_α arrays `xca`, `yca`, `zca`, `betaca`, `segnameca`, and `pdbaindexca`.

(b) `SUBROUTINE pdbreadDr(filenm,fileInt,alphaInt,num)` - Same as `pdbread` accept for the corresponding structure pertaining to the allosteric calculations, i.e. the secondary structure for the allosteric calculation.

10. `psfread.f90`

(a) `SUBROUTINE getnum(filenm,fileInt,istart,num)` - Returns the number of atoms referenced in the PSF file and the corresponding line in the PSF file that the atom entries start on.

(b) `SUBROUTINE psfread(filenm,fileInt,alphaInt,istart,num)` - Reads the PSF file and populates the arrays `aindex`, `residc`, `resname`, `atomname`, `atomtype`, `charge`, `mass`, and `resid`.

(c) `SUBROUTINE readDQ(filenm,fileInt)` - Reads the file with the Δq charges for the donor/acceptor calculation.

(d) `SUBROUTINE eofile(fileInt,fileName,eofInt)` - Returns the number of lines in a file, `eofInt`.

---

[1] D. R. Martin, S. B. Ozkan, and D. V. Matyushov, Phys. Biol. **9**, 036004 (2012).
[2] D. R. Martin and D. V. Matyushov, J. Chem. Phys. **137**, 165101 (2012).