NAME

carma − A molecular dynamics analysis program

SYNOPSIS

carma [options] [<PSF> and <DCD> files]

I HATE MANUALS

Tell me about it. Anyway, here is a quick tutorial for the impatient: go to a GNU/linux box, open a unix shell, install the program, create a clean directory containing your DCD and PSF files and make this your working directory. From now on, I will assume for simplicity that your DCD −PSF system contains one polypeptide chain with a segment identifier equal to `"A"`, plenty of water molecules, and some sodium and chloride ions. For the purpose of this tutorial, grab also a copy of your protein−only PSF file (ie. a PSF file containing only the protein atoms as produced, for example, from PSFGEN). To make things go fast for this tutorial, don't start with your brand new 90 Gbytes − 100,000 atoms DCD file (but you do need a sufficiently large number of frames for the statistical analyses to be meaningful).

QUICK VIEW OF TRAJECTORIES

To start with, let's have a quick look at the trajectory to make sure everything looks OK:

carma my.dcd my.psf

and press ENTER a couple of times. When you had enough, make the shell window active again and stop the program with CTRL −C. If you haven't had enough, see the section entitled **Graphics tricks and examples**.

RMSD MATRIX

Good, now that you had enough, let's calculate something useful:

The aim is to produce a matrix containing the rms deviations (using CA atoms only) between all possible pairs of structures from your trajectory. The problem is that if your trajectory contains, say, 100,000 frames, then your matrix will end −up being a 38 GBytes file which −even if you can calculate it− will be useless for any further analysis. So, instead of using each and every frame, use every, say, 500th structure to reduce its size:

carma −verbose −cross −step 500 −segid A my.dcd my.psf

A new file will be produced in your current directory named **carma.RMSD.matrix** which contains the matrix. Prepare a colour postscript representation of the matrix with

carma −colour − < carma.RMSD.matrix

and view the resulting **carma.stdin.ps** file with your favorite postscript viewer. The color gradient goes from dark blue (for small values), through yellow (for medium values), to dark red (for large values), with the origin at the upper left −hand corner.

DIHEDRAL PCA

Let's move−on to principal component analysis:

carma −verbose −write −color −segid A −dPCA 5 3 298 my.dcd my.psf

This will do a dihedral angle (phi,psi) principal component analysis of your protein's trajectory. Because dPCA uses internal coordinates, there is no need to remove overall translations and rotations from beforehand. The three numbers following the **−dPCA** flag are the important ones: '5' is the number of principal components which will be written to a file (in the current directory) named **carma.dPCA.fluctuations.dat**, '3' is the number of principal components for which DeltaG (energy landscapes) postscript diagrams will be produced, and finally, '298' is the temperature of your simulation in Kelvin. If you list the directory contents (after carma is finished) you should see quite a number of files:

```
carma.clusters.dat              carma.dPCA.eigenvalues.dat
carma.dPCA.DG_01_02.dat         carma.dPCA.eigenvectors.dat
```

```
carma.dPCA.DG_01_02.ps          carma.dPCA.fluctuations.dat
carma.dPCA.DG_01_03.dat         my.dcd
carma.dPCA.DG_01_03.ps          my.dcd.dPCA.varcov.dat
carma.dPCA.DG_02_03.dat         my.dcd.dPCA.varcov.ps
carma.dPCA.DG_02_03.ps          my.psf
```

my.psf and **my.dcd** are where we started from, **carma.dPCA.eigenvalues.dat** and **carma.dPCA.eigenvectors.dat** are exactly what you think they are (eigenvectors are in columns, top−first), **carma.dPCA.fluctuations.dat** has already been described, **my.dcd.dPCA.varcov.dat** and **my.dcd.dPCA.varcov.ps** are the variance−covariance matrix and corresponding postscript plot, and finally, the various **carma.dPCA.DG_*** files are the data and postscript plots of the free energy landscapes on the principal component planes defined by the eigenvectors pairs 1−2, 1−3 and 2−3. Use your postscript viewer to examine the images, use your plotting program (xmgr, gnuplot, grace, ...) to examine the data files, etc.

If you want to include the side−chains' chi1 angles in the analysis, repeat the calculation this time adding the **−chi1** flag in the command line.

DIHEDRAL−PCA−BASED CLUSTER ANALYSIS

The file **carma.clusters.dat** contains the results from a cluster analysis performed using the top three principal components and is described in detail in the section entitled **PCA−based cluster analysis**. Here we will only deal with the practical part. We start by examining how well (or otherwise) the cluster analysis went: use awk to isolate the frames that carma thinks that belong to the most populated cluster (corresponding, hopefully, to the major conformer):

```
awk '{if ($2==1) print $1,$3,$4,$5}' carma.clusters.dat > C_01.dat
```

and use your plotting program to make a scatter plot of the 2nd and 3rd columns of the **carma.dPCA.fluctuations.dat** file and **superimpose** on this (using a different colour) a scatter plot of the 2nd and 3rd columns of the **C_01.dat** file. If all went well you should see the second plot overlapping the most prominent cluster of the first plot.

DPCA CLUSTERS: FITTING, COVARIANCE MATRICES & AVERAGE STRUCTURES

We know the trajectory frames that constitute the core of the largest cluster, but this is not enough. Let's examine this cluster in more detail: first prepare a new DCD file containing just the frames corresponding to this cluster (note that the various carma flags can be shortened to their uniquely identifiable minimum):

carma −v −sort C_01.dat my.dcd

The new DCD file is named **carma.reordered.dcd**. To calculate an average structure, we must remove overall rotations and translations. While we do this, we might as well get rid of all those waters and ions and keep just the protein atoms:

carma −v −w −fit −atmid ALLID −segid A carma.reordered.dcd my.psf

The resulting DCD file is named **carma.fitted.dcd** and contains the coordinates of the protein component only, for the first cluster only, and with overall rotations and translations removed. As a bonus you also get a file named **carma.fit−rms.dat** containing (in its second column) the rms deviation of each frame from the first frame (you can of course use your plotting program to see the corresponding graph). Confirm that everything looks OK and rename the DCD file to something more useful:

carma carma.fitted.dcd protein.psf
```
mv carma.fitted.dcd cluster_01.dcd
```

where **protein.psf** is the PSF file of the protein component only (as produced, for example, from

PSFGEN). We can now calculate the normalised variance −covariance matrix and the average structure in the form of a PDB file, plus a PDB file containing a superposition of representative structures belonging to this cluster. The following line will do the trick:

carma −v −w −col −cov −dot −super −atmid HEAVY cluster_01.dcd protein.psf

Use your postscript viewer to look at the covariance matrix in the file **cluster_01.dcd.varcov.ps** and your favorite molecular graphics program to see the structures in the files **carma.superposition.pdb** and **carma.average.pdb**. Use your plotting program to view the rms deviations from this average structure as contained in the file **carma.rms−average.dat**. Since you are here, a possibly more useful covariance matrix to look at, is the normalised per residue (in reality, per CA) map:

carma −v −w −col −cov −dot −norm cluster_01.dcd protein.psf

(note that the new average structure produced by the program will this time contain only CA atoms). Because **carma**'s default is to use to CA atoms, there is no need to say **−atmid CA** as you might have expected.

FITTING, REMOVAL OF WATERS AND IONS

Enough with dihedral PCA, let's start with Cartesian PCA: delete (or copy and then delete) everything but the files my.dcd, my.psf and protein.psf, and then give:

carma −v −w −fit −atmid ALLID −segid A my.dcd my.psf
`mv carma.fitted.dcd protein.dcd`

The line above will superimpose all atoms of segid A of each and every frame on the first frame of the trajectory and will produce a new DCD file name **carma.fitted.dcd** containing the coordinates (of only all atoms of segid A, no waters or ions) with rotations and translations removed. You will also get a file named **carma.fit−rms.dat** containing in its first two columns the rms deviation from the reference structure vs. frame number [the rest of the columns in the carma.fit−rms.dat file are the rotation matrix (9 columns) and the translation vector (3 columns)]. If you want to use a different reference frame, use the **−reference** flag. If you want to use selected atoms for fitting, see the flag **−index**.

CARTESIAN PCA

Now that we have removed rotations and translations, we can go ahead with cartesian PCA. Before that, however, go to the section entitled **Appendix: Preparation of CA −only PSF files** and prepare a file named CAs.psf for the CA −only atoms of your protein (the reason for using only the CA atoms for the this section is to keep matrices' sizes reasonably small). Now do:

carma −v −w −col −cov −eigen −proj 5 3 298 protein.dcd protein.psf

where **protein.psf** is the PSF file of the protein component only (as produced, for example, from PSFGEN). The line above will do a cartesian principal component analysis of your protein's trajectory **using only the CA atoms**. If you would rather use, say, all non −hydrogen atoms (and have enough CPU time to spare), add an **−atmid** flag in the spirit of e.g. '−atmid HEAVY'. The three numbers following the **−proj** flag have the following meaning: '5' is the number of principal components which will be written to a file (in the current directory) named **carma.PCA.fluctuations.dat**, '3' is the number of principal components for which DeltaG (energy landscapes) postscript diagrams will be produced, and finally, '298' is the temperature of your simulation in Kelvin. If you list the directory contents you will something like this:

```
carma.PCA.DG_01_02.dat          carma.PCA.fluctuations.dat
carma.PCA.DG_01_02.ps           carma.clusters.dat
carma.PCA.DG_01_03.dat          carma.fit-rms.dat
carma.PCA.DG_01_03.ps           protein.dcd
carma.PCA.DG_02_03.dat          protein.dcd.varcov.dat
carma.PCA.DG_02_03.ps           protein.dcd.varcov.ps
```

```
my.dcd                          carma.PCA.eigenvalues.dat
my.psf                          carma.PCA.eigenvectors.dat
protein.psf                     CAs.psf
```

my.psf, **my.dcd**, **protein.psf** and **CAs.psf** are where we started from, **carma.PCA.eigenvalues.dat** and **carma.PCA.eigenvectors.dat** are exactly what you think they are (eigenvectors are in columns, top−first), **carma.PCA.fluctuations.dat** has already been described, **protein.dcd.varcov.dat** and **protein.dcd.varcov.ps** are the variance−covariance matrix and corresponding postscript plot, and finally, the various **carma.PCA.DG_*** files are the data and postscript plots of the free energy landscapes on the principal component planes defined by the eigenvectors pairs 1−2, 1−3 and 2−3. Use your postscript viewer to examine the images, use your plotting program (xmgr, gnuplot, grace, ...) to examine the data files, etc.

CARTESIAN−PCA−BASED CLUSTER ANALYSIS

This is more−or−less identical with the dPCA cluster analysis: The file **carma.clusters.dat** contains the results from a cluster analysis performed using the top three principal components and is described in detail in the section entitled **PCA−based cluster analysis**. Here we will only deal with the practical part. Start by examining how well (or otherwise) the cluster analysis went: use awk to isolate the frames that carma thinks that belong to the most populated cluster (corresponding, hopefully, to the major conformer):

```
awk '{if ($2==1) print $1,$3,$4,$5}' carma.clusters.dat > C_01.dat
```

and use your plotting program to make a scatter plot of the 2nd and 3rd columns of the **carma.PCA.fluctuations.dat** file and **superimpose** on this (using a different colour) a scatter plot of the 2nd and 3rd columns of the **C_01.dat** file. If all went well you should see the second plot overlapping the most prominent cluster of the first plot.

CARTESIAN PCA CLUSTERS: FITTING, COVARIANCE MATRICES & AVERAGE STRUCTURES

We know the trajectory frames that constitute the core of the largest cluster, but this is not enough. Let's examine this cluster in more detail: first prepare a new DCD file containing just the frames corresponding to this cluster:

carma −v −sort C_01.dat protein.dcd

The new DCD file is named **carma.reordered.dcd**. Although the protein.dcd file has rotations and translations removed, it is probably safer to re−align all cluster frames once more (using again only CA atoms):

carma −v −w −fit −segid A carma.reordered.dcd protein.psf

The resulting DCD file is named **carma.fitted.dcd** and contains the coordinates of only the CA atoms of the protein component, for the first cluster only, and with overall rotations and translations removed. As a bonus you also get a file named **carma.fit−rms.dat** containing (in its second column) the rms deviation of each frame from the first frame (you can of course use your plotting program to see the corresponding graph). Confirm that everything looks OK and rename the DCD file to something more relevant:

carma carma.fitted.dcd CAs.psf

```
mv carma.fitted.dcd cluster_01.dcd
```

where **CAs.psf** is the PSF file of the CA atoms of the protein component (see section **Appendix: Preparation of CA−only PSF files** for guidance in producing such a PSF file). We can now calculate the normalised variance−covariance matrix and the average structure in the form of a PDB file, plus a PDB file containing a superposition of representative structures belonging to this cluster. The following lines will do the trick:

carma −v −w −col −cov −dot −super cluster_01.dcd CAs.psf

Use your postscript viewer to look at the covariance matrix in the file **cluster_01.dcd.varcov.ps** and your favorite molecular graphics program to see the structures in the files **carma.superposition.pdb** and **carma.average.pdb** (which will contain the CA atoms only). Use your plotting program to view the rms deviations from this average structure as contained in the file **carma.rms−average.dat**. Since you are here, a possibly more useful covariance matrix to look at, is the normalised per residue (in reality, per CA) map:

carma −v −w −col −cov −dot −norm cluster_01.dcd CAs.psf

'SEEING' THE MOTION CORRESPONDING TO A SELECTED PRINCIPAL COMPONENT

Do a 'more carma.PCA.DG_01_02.ps' and note the limits of the plot reported (as a comment) in the header of the postscript file. Then using the image **carma.PCA.DG_01_02.ps** and these limits, determine the minimum and maximum observed values of the first principal component. If you hate using rulers and pens, use awk:

```
awk '{print $2}' carma.PCA.fluctuations.dat | sort -n | tail -1
awk '{print $2}' carma.PCA.fluctuations.dat | sort -n | head -1
```

Now, use these two numbers to run **carma** (this example again assumes that you only care about the CA atoms):

carma −verb −write −col −cov −eigen −play 1 −4 5 cluster_01.dcd CAs.psf

where the numbers '−4' and '5' are those you determined from the awk lines above. A new DCD file has been produced named **carma.play.dcd**. Play it using for example VMD (vmd carma.play.dcd CAs.psf) or carma (carma carma.play.dcd CAs.psf). Finally, prepare a PDB file showing the motion due to the first eigenvector:

carma −v −w −pdb −last 10000 −step 500 carma.play.dcd CAs.psf

```
cat carma.play.dcd.*.pdb > eigen_01.pdb
rm carma.play.dcd.*.pdb
```

where the number after −last (10000 in this example) is one half of the total number of frames in the carma.play.dcd file.

SOLUTE ENTROPY CALCULATION

Delete (or copy and then delete) everything but the files my.dcd, my.psf, protein.psf and CAs.psf. Then, remove rotations−translations

carma −v −fit −atmid ALLID −segid A my.dcd my.psf

and start calculating entropies using, for example, increasing number of frames (the number '320' next to '−temp' is the simulation's temperature in Kelvin) :

carma −v −cov −eigen −mass −temp 320 −atmid ALLID −segid A −last 1000 carma.fitted.dcd protein.psf

carma −v −cov −eigen −mass −temp 320 −atmid ALLID −segid A −last 2000 carma.fitted.dcd protein.psf

carma −v −cov −eigen −mass −temp 320 −atmid ALLID −segid A −last 3000 carma.fitted.dcd protein.psf

...

PER FRAME DISTANCE MAPS

To calculate coloured CA−CA distance maps for the structures corresponding to frames 1, 1001, 2001, ..., give:

carma −v −w −col −step 1000 my.dcd my.psf

keeping in mind that, depending on the number of frames you have, this may produce quite a number of files. If you want distance maps for all non −hydrogen atoms you will have to define both −atmid & −segid (otherwise you'll pick−up all water oxygens as well):

carma −v −w −col −atmid HEAVY −segid A −step 5000 my.dcd my.psf

AVERAGE DISTANCE MAPS & RMS DEVIATION FROM THEM

Calculate (i) the average distance map for all non −hydrogen atoms, and, (ii) a map depicting the rms deviation of these distances from their average:

carma −v −w −col −rms −atmid HEAVY −segid A my.dcd my.psf

View the postscript files **my.dcd.averag.ps** and **my.dcd.rmsdev.ps**, and then merge them about their diagonal:

carma my.dcd.averag.ps my.dcd.rmsdev.ps

You now have a new postscript file named **carma.merged.ps** which contains one half from the average distance map, the other half from the rmsd map.

Similarly, say that you want to compare the rms deviations of the average CA −CA distance maps between the first and the second half of the trajectory (assumed to have 40000 frames). Here it is:

```
carma -v -w -col -rms -mrms 4 -last 20000 my.dcd my.psf
mv my.dcd.rmsdev.ps 1.ps
carma -v -w -col -rms -mrms 4 -first 20001 my.dcd my.psf
mv my.dcd.rmsdev.ps 2.ps
carma 1.ps 2.ps
```

and then view the results in the postscript file **carma.merged.ps** (the −**mrms** flag in the previous lines has been added to guarantee a constant scale between the two graphs).

EXTRACTING SELECTED PDB FILES

Get a series of PDB files containing all atoms of segid A for the frames 1, 1001, 2001, ... :

carma −v −pdb −atmid ALLID −segid A −step 1000 my.dcd my.psf

This will produce a series of PDB files of the form **my.dcd.??????.pdb)** where the question marks are substituted by the corresponding frame number. Please see section **Bugs & features** for an important feature concerning atom names and segment identifiers.

If you need a specific frame, the following line should do it:

carma −v −pdb −atmid ALLID −segid A −first 16384 −last 16384 my.dcd my.psf

RADIUS OF GYRATION, DISTANCES ANGLES & TORSIONS

Calculate the mass−weighted radius of gyration of segid A, and place the results in the file **carma.Rgyration.dat**:

carma −v −rg −atmid ALLID −segid A my.dcd my.psf

Calculate the distance between the 3rd and the 48th atom (the atom numbers being those reported in the PSF file) and place the results in the file **carma.distances**:

carma −v −dist 3 48 −atmid ALLID my.dcd my.psf

Calculate the torsion angle (in the range −180 to +180 degrees) defined by the atoms 12, 34, 52 and 98, and place the results in the file **carma.torsions**:

carma −v −torsion 12 34 52 98 −atmid ALLID my.dcd my.psf

Calculate the bending angle (in the range 0 to 180 degrees) defined by the atoms 23, 36 and 48, and place the results in the file **carma.bendangles**:

carma −v −bend 23 36 48 −atmid ALLID my.dcd my.psf

DESCRIPTION

Carma is light−weight command−line−driven molecular dynamics analysis program for DCD−type trajectory files. **Carma** started as a CA−CA distance map calculator, but ended supporting most of the steps required for a principal component analysis of molecular dynamics trajectories (also known as essential dynamics analysis). In recent versions it also acquired the ability to display (on X11−capable machines) a trajectory. Nevertheless, the initial features of the program are still there and so you can still use the program for something as simple as making a postscript file containing a grayscale image of the CA −CA distance map of your protein. This mixing of seemingly uncorrelated calculations does have its cost : it is almost impossible to remember what combination of keywords will do what (which, reassuringly, is the case with most unix commands).

For the discussion that follows I will be referring to the CA atoms only of the whole molecule which is the default atom selection for the program. Note, however, that it is possible to select other atom types or SegIDs (SEGment IDentifiers). See keywords −atomid and −segid.

COMPILING THE PROGRAM

Have a look in the bin/ subdirectory of the distribution (just in case there is an executable already available for your machine).

To build from source you will need some familiarity with compilers and libraries. **Carma** dependencies are:

* For the eigen calculation the LAPACK library is needed (possibly with BLAS).

* For graphics support you need the Ygl v.4.1 library (google for 'Ygl').

Compiling the program per se is easy. But linking with the LAPACK library may require the usage of a fortran compiler. For example, on a GNU/Linux machine :

gcc −DGRA −D_FILE_OFFSET_BITS=64 −Wall −pedantic −O −c carma.c

g77 −DGRA −D_FILE_OFFSET_BITS=64 −Wall −pedantic −O carma.o −L/usr/X11R6/lib −lYgl −lXext −lX11 −llapack −lblas −lm

It is possible to avoid the separate link step if you know which libraries the fortran compiler needs. For example, on an OSF machine you could use something like the following (but please note that I no longer have access to an alpha machine, so the following line has not been tested) :

cc −DGRA −newc −fast −arch generic −tune generic carma.c −lYgl −lXext −lX11 −llapack −lblas −lm −lUfor −lfor −lFutil −lm −lots −lc

Clearly, this short description isn't very helpful −−−and will probably fail−−− if you are not using a GNU/Linux or OSF machine. Fear not : I hearby undertake to help you with your compilation (and/or linking) problems for other architectures (excluding windoze for which an executable is already available) as long as you agree to provide me with a copy of an executable for the next **carma** version (so that I can include it with the distribution).

GENERAL OPTIONS

All input to the program is through (case −insensitive) command−line options. Abbreviated (but uniquely identifiable) forms of the various flags are also valid.

−VERB

Tells **carma** to stop being so esoteric about what it is doing.

−WRITE

For each and every postscript file written by the program, an additional file with the suffix **.dat** will also be produced. This will contain the results of the respective calculation performed.

−SIGMA <FLOAT>

Apply a sigmoidal function weighting to all data written out by the program. This is intended as a means to increase the contrast of the cross−correlation matrices. The function used is 2.0/[1+exp(Sigma*x)] − 1.

−PDB

Write−out the selected atoms from the selected frames to a series of PDB files. The names of the PDB files are constructed using the name of the DCD file plus the frame number. See section **Bugs & features** for the treatment of long atom and segid names.

ATOM, SEGMENT AND FRAME SELECTION

−ATOMID <STRING>

carma's default is to use only the CA atoms for its calculations. This option allows the additive selection of other atom types. For, example the combination

```
-atomid CA  -atomid C  -atomid N  -atomid O
```

will select the protein backbone atoms. See also the **−segid** option below. You can select all atoms using the special selection

```
-atomid ALLID
```

or, all non−hydrogen atoms using the expression

```
-atomid HEAVY
```

−SEGID <STRING>

carma's default is to use all segments for its calculations. This option allows the additive selection of specific segments that should be used. For, example the combination

```
-segid A  -segid D  -atomid CB
```

will perform subsequent calculations using only the coordinates of the CB atoms that belong to the A or D segments. See also the **−atomid** option above. The **−segid** option is not valid for the PDB−only mode of the program.

−FIRST <POSITIVE INTEGER>
−LAST <POSITIVE INTEGER>
−STEP <POSITIVE INTEGER>

These three keywords define the DCD frames that will be used for subsequent calculations. The default is to use all frames present in the DCD file, until the end −of−file is reached (the value for the number of coordinate sets which is given in the header of the DCD files is ignored).

carma completely ignores the information about timesteps (starting and interval) from the DCD header and uses the following reference scheme : the first frame contained in the DCD file is frame number 1, the second is number 2, and so on until the end of the DCD file. The convention for the **−step** keyword is : If step=1, then each and every frame will be used. If step=2 then the first frame will be used, the next skipped, the third used, etc.

−NOCAS

This flag instructs the program to produce DCD files containing coordinates not only for the selected atoms, but for all atoms that were present in the input DCD file. **Note well : all non−selected atoms will be copied on an 'as −is' basis to the output file**. The implication is that any all−atom representation calculated on from the newly produced DCD file will be

meaningless. The only use for this keyword (that I can envision) is saving you from producing a PSF file for the atoms you selected. In summary, this is a useless and possibly confusing keyword. Don't use it.

REMOVING ROTATIONS – TRANSLATIONS

−FIT

In this mode **carma** will produce two files. The first is a DCD file named **carma.fitted.dcd** which will contain a trajectory with the global rotations and translations removed. This is based on the application of Kabsch's algorithm to least −squares fit all frames onto the first frame (or, through the −**ref** keyword, a user−defined reference frame). Which atoms will be used for the fitting (and will, thus, be contained in the DCD) is determined by the −**segid** and −**atomid** flags. You can explicitly select a subset of those using the −**index** keyword. The second file (**carma.fit−rms.dat**) is an ASCII file containing (in its second column) the rms deviations of each structure versus the reference structure (plus a series of columns containing the rotation and translation matrices).

−REF <POSITIVE INTEGER>

This keyword defines the trajectory frame that will be used as reference for removing global rotations and translations. It only makes sense in the presence of the −**fit** keyword.

−INDEX

If the structure contains flexible parts, you may want to ignore them during the least −squares fitting procedure. When this keyword is present, **carma** will attempt to open (from the current directory) a formatted ASCII file (text file) with the name **fit.index** containing the **one−based** indeces of the atoms to be used for least −squares fitting (one index number per line). The selection applies after the −**segid** and −**atomid** selections have taken place. To make this clear: if your original DCD file contains coordinates for 40000 atoms, but you have selected (through the −**segid** and −**atomid** flags) 5000 atoms, then the **fit.index** file should contain index numbers ranging from 1 to 5000. The trajectory file created by the program will contain all selected atoms and not just those used for the fitting (referring to the previous example, the resulting DCD file will contain coordinates for 5000 atoms (and not just those whose indeces were present in the **fit.index** file)).

−NOFIT

This keyword allows you to calculate the rms deviation between the starting (or reference) structure and those that follow, but **without** least−squares superimposing them (before calculating the rmsd). We came across the need for such a calculation when studying a homodimer: we run carma once to align all frames using one of the monomers, and then used it again (with the −nofit keyword) to calculate rms for the other (and using the DCD file produced from the first run of carma).

RMSD MATRICES

−CROSS

This keyword allows you to calculate the rms deviation between all possible pairs of structures from your trajectory. The resulting two−dimensional matrix will be written in the file **carma.RMSD.matrix** in the current directory. You will need to define a value for the −**step** flag to keep the matrix size manageable. Atom and SEGID selections apply as usual. You can produce a colour−postscript representation of the resulting matrix again with carma (see the tutorial at the beginning of this document).

FRACTION OF NATIVE CONTACTS

−QFRACT <CUTOFF> <RESIDUE SEPARATION>

Recent versions of carma can calculate native contacts. The assumption is that the first frame of the trajectory is the 'native' structure (use, say, catdcd to do this), and all calculations are performed with respect to this (first) frame. The two numbers after the flag are (i) the distance cutoff (in Angstrom) for the contacts, and, (ii) the separation (in number of residues) required for a

contact. If the second number is 3, then the difference in residue numbers must be greater (but not equal) to 3. Carma will create a file named **carma.Qfraction.dat** containing three similarity measures. These are Q, Qs and q (in this order) as defined in the supplementary information of the Cho, Levy and Wolynes (2005), PNAS, 103, 586 −591 paper (http://www.pnas.org/content/103/3/586) with the difference that for the calculation of q the residue separation (given in the command line) is being applied.

PRINCIPAL COMPONENT ANALYSIS (ESSENTIAL DYNAMICS)

[1] CARTESIAN VARIANCE −COVARIANCE, AVERAGE STRUCTURE, PCA & ENTROPY

−COV

This signals the calculation of the variance −covariance matrix for the coordinates contained in the trajectory file. This keyword must be present for any of the cartesian PCA keywords that follow to apply.

−MASS

Calculate the mass −weighted variance −covariance matrix. The 163 atomic types known to the program are (not surpisingly) hard coded (you can find them near the top of the source code). Still, if an unknown atomic type is encountered, carma will try to grab its mass from the PSF file.

−DOT

carma will by default calculate the variance −covariance matrix of the Cartesian displacements of the individual x,y,z coordinates of all selected atoms (CAs by default). This keyword allows you to calculate the variance −covariance matrix of the atoms' fluctuations by taking the dot product between the displacement vectors. Note that this option is not valid when performing the cartesian principal component analysis. A useful by −product of this calculation is that a PDB file (**carma.average.pdb**) containing the average structure of all selected atoms will be produced. The B −factor column in this PDB file will contain the atomic root −mean squared fluctuations (in Angstrom). Additionally, a file named **carma.rms −average.dat** will be produced which will contain the rms deviation of each frame from this average structure. **Please note** that the calculation of this rmsd value does **not** involve a least −squares superposition of the average structure with the trajectory structures.

−SUPER

When calculating the average structure, also prepare a PDB file named **carma.superposition.pdb** which will contain a superposition of 500 equally spaced structures from the trajectory. The B −factor columns for all structures will contain RMSFs.

−NORM

Instructs the program to output the normalised (cross −correlation) matrix instead of the raw variance −covariance matrix. Again, this option is not valid when performing the principal component analysis.

−EIGEN

If you use the executables provided with the distribution (or if you have the LAPACK library to link the program with), **carma** will be able to calculate and write out for you the (sorted) eigenvalues and eigenvectors of the raw variance −covariance matrix (but not the cross correlation matrix). The results will be written to the files **carma.eigenvectors.dat** and **carma.eigenvalues.dat**.

−PROJ <INTEGER(1)> <INTEGER(2)> <TEMP> [<SIGMA_CUTOFF>]

This keyword only applies if both **−cov** & **−eigen** have been defined. It instructs the program to calculate and output a formatted ASCII file (with the name **carma.fluctuations.dat**) containing the top **<integer(1)>** principal components (projections of the fluctuations along the top **<integer(1)>** eigenvectors). It will then prepare DeltaG (free energy landscape) postscript plots for all combinations of the top **<integer(2)>** eigenvectors. In order for these plots to have units of

kcal/mol, it is necessary to define the temperature for the simulation <**temp**>. The postscript files produced will be named carma.PCA.DG _01_02.ps, carma.PCA.DG _01_03.ps, ..., depending on the vectors used for them. Finally (and if the second argument <integer(2)> is greater than 2), the program will perform a simple−minded cluster analysis on the space defined by the three eigenvectors corresponding to the three largest eigenvalues. The results from this calculation are placed in a file named **carma.clusters.dat** and are discussed in the section entitled **PCA−based cluster analysis**. The last optional argument <**sigma_cutoff**> determines a cutoff for the cluster determination procedure. By default its value is determined automatically, and is discussed in the section **PCA−based cluster analysis**.

−OUT <FIRST> <LAST> <STEP>

This keyword only applies if both **−cov** & **−eigen** have been defined. It instructs the program to output a DCD file with the name **carma.proj.dcd** which contains the projection of the input trajectory on the specified eigenvectors.

−PLAY <EIGENVECTOR> <MAX AMPLITUDE> <MIN AMPLITUDE>

This keyword only applies if both **−cov** & **−eigen** have been defined. It instructs the program to output a DCD file with the name **carma.play.dcd** which contains a smooth representation of the motion due to one (selected) eigenvector. The maximum and minimum amplitudes of the fluctuations along this eigenvector must also be specified and these usually come from a previous run of the program with the **−proj** flag.

−ARTIFICIAL <VECTORS> <FRAMES>

This keyword only applies if both **−cov** & **−eigen** have been defined. With this keyword **carma** can be used to produce a DCD file (with the name **carma.arti.dcd**) containing a set of structures that are derivable from a set of eigenvectors and eigenvalues (obtained from a principal component analysis of the given trajectory). To perform the calculation the program needs the fluctuations contained in the file **carma.fluctuations.dat** (assumed to be present in the current directory from a previous run of the program with the flag **−proj**). To make things clear : if you want to prepare a set of 10000 structures derivable from the first 30 eigenvactors, you should run **carma** twice : the first time including the flag **−proj 1 30 1**, the second time including the flag **−arti 30 10000**.

−USE

This flag instructs the program to read previously calculated eigenvalues and eigenvectors instead of re−calculating them. These will be read from the files **carma.eigenvectors.dat** and **carma.eigenvalues.dat** which must be present in the current directory. Using previously calculated values does lead to some loss of precision (due to the format of these ASCII files), but for large problems (where the eigen−calculations are the heaviest part of the calculation) it may be the preferred choice. For a note on the precision and accuracy aspects of the eigen−calculation see the **Bugs** section below.

−TEMPERATURE <FLOAT>

The presence of the **−temp** flag (followed by the temperature in Kelvin that the simulation was performed) instructs the program to calculate the solute's entropy using the eigenvalues of the mass−weighted variance−covariance matrix. The presence of the −cov, −eigen and −mass flags is required. The entropy is calculated using two different formulas (reported in the program output). For the calculation to be meaningful, you should select all solute atoms (and not just the default CAs). Also note that the entropy value returned by the program (in J/molK) assumes that your trajectory file contains coordinates for only one copy of the molecule of interest. If this is not true, you will have to divide the entropy value returned by carma with the number of molecules.

−3D

When this flag is present, carma will produce two additional files named **carma.3d_landscape.cns** and **carma.3d_landscape.na4**. These are three−dimensional map files containing the distribution of the top three principal components, which may be viewed as

three−dimensional folding landscapes. The **carma.3d_landscape.cns** can be loaded and viewed directly with VMD (you will have to adjust the cutoff for the various isosurfaces). The **carma.3d_landscape.na4** is for use with the CCP4 suite of programs.

PRINCIPAL COMPONENT ANALYSIS (ESSENTIAL DYNAMICS)

[2] DIHEDRAL ANGLE (PHI −PSI) VARIANCE −COVARIANCE & PCA

−DPCA <INTEGER(1)> <INTEGER(2)> <TEMP> [<SIGMA_CUTOFF>]

This flag signals the beginning of the dihedral PCA (dPCA) analysis. For this to work you will have to use the **−segid** flag to define at least one protein chain that will be used for the analysis. The program will automatically select all necessary main −chain atoms, will build a list of psi −phi angles that will be used for the analysis, will calculate the variance −covariance matrix (file my.dcd.dPCA.varcov.ps), and calculate its eigenvectors and eigenvalues (which will be written−out in the files carma.dPCA.eigenvalues.dat and carma.dPCA.eigenvectors.dat). It will then prepare a file named **carma.dPCA.fluctuations.dat** containing the top **<integer(1)>** principal components (projections of the fluctuations along the top **<integer(1)>** eigenvectors). It will then prepare DeltaG (free energy landscape) postscript plots for all combinations of the top **<integer(2)>** eigenvectors. In order for these plots to have units of kcal/mol, it is necessary to define the temperature for the simulation **<temp>**. The postscript files produced will be named carma.dPCA.DG_01_02.ps, carma.dPCA.DG_01_03.ps, ..., depending on the vectors used for them. Finally (and if the second argument <integer(2)> is greater than 2), the program will perform a simple−minded cluster analysis on the space defined by the three eigenvectors corresponding to the three largest eigenvalues. The results from this calculation are placed in a file named **carma.clusters.dat** and are discussed in the section entitled **PCA−based cluster analysis**. The last optional argument **<sigma_cutoff>** determines a cutoff for the cluster determination procedure. By default its value is determined automatically, and is discussed in the section **PCA−based cluster analysis**.

−CHI1

This flag tells carma to include in the dihedral PCA analysis the side −chains' chi1 angles. The program will report the number of phi, psi and chi1 angles used for the analysis. Carma has fixed ideas about the names of the atoms needed to define all required angles and will fail if the atoms with names N, CA, C, CG, CG1, OG, OG1, SG and CB do not suffice for defining the required angles.

ION & WATER DISTRIBUTION MAPS

−MAP <FLOAT(1)> ... <FLOAT(7)>

This keyword allows the calculation of an average distribution map (of waters, ions, ...) during the trajectory. The map is written in a file named **carma.fitted.dcd.map** using the **xplor** and **cns** electron density map format (it may be necessary to rename the map file so that its suffix is .edm). The first six numbers after the flag are the limits (in the orthogonal frame and in Angstrom) along x, y and z that the map will occupy. The last number is the grid spacing (again in Angstrom). The distribution map will be calculated for all selected atoms. A typical usage would be :

carma −v −map −30 30 −20 20 −15 20 0.50 −atmid OH2 wat_ions.psf wat_ions.dcd

Please do note the following :

* The method will fail if the solute diffused away from the water box. The reason for this is that the periodic boundary periodicity is not being taken into account.

* The method will fail if the solute does not have a well behaved average structure.

* The calculation assumes that your DCD file is such that rotations and translations have been removed from the solute (but, of course, the atomic species for which you want to calculate the distribution map must still be present in the trajectory). You can do the trick using **carma** with something in the spirit of

```
carma -v -fit -index -atmid OH2 -atmid SOD -atmid CLA -atmid CA my.dcd my.psf
```

where the fit.index file (which should be present in the current directory) should contain (the one−based) indeces for the CA atoms (after the selection imposed by the −atmid cards).

−FMAP <FLOAT(1)> ... <FLOAT(7)>

The problem with the **−map** flag is that it necessitates the creation of a large intermediate trajectory file. The **−fmap** flag attempts to speed things−up by calculating the distribution map on the fly. The meaning of the seven parameters following **−fmap** are the same as with the **−map** flag. The difference is that **−fmap** must be used simultaneously with the **−fit** and **−index** flags and requires the creation of two index files. The first (file **fit.index**) will contain the indeces for the solute atoms that will be used to remove overall rotations and translations. The second index file (**fmap.index**) should contain the indeces of the atoms for which the distribution map will be calculated (eg. water oxygens). Both index files should be one −offset (the first atom has index 1 and not 0). Once you have the index files, use the program with something in the spirit of :

carma −v −fit −index −atmid OH2 −atmid CA −fmap −30 30 −20 20 −15 20 0.50 my.psf my.dcd

Please note that the indeces (in the index files) should refer to atom indeces **following** all selections based on **−atmid** and **−segid** cards. To make this clear: lets say that your original DCD file contains 40000 atoms, of which 3000 are solute atoms (with 300 CA atoms), 36999 are water atoms, and there is one sodium ion. If you select only the OH2 and CA atoms (as in the previous example), then **carma** will only "see" 12633 atoms (300 CA atoms plus (36999/3)=12333 OH2 atoms). The two index files should contain numbers (indeces) ranging from 1 to 12633. Most probably, you would want to prepare a **fit.index** file containing the numbers from 1 to 300 (which would tell carma to remove rotations/translations using the CA atoms), and a **fmap.index** file containing numbers from 301 to 12633 (which would tell carma to calculate a density distribution map for water oxygens).

Please also note the following :

* The method will fail if the solute diffused away from the water box. The reason for this is that the periodic boundary periodicity is not being taken into account.

* The method will fail if the solute does not have a well behaved average structure.

RADIUS OF GYRATION, MOLECULAR SURFACE AREA, DISTANCES & TORSIONS

−RG

This will calculate the mass−weighted radius of gyration of the selected atoms. To use this flag, you must also define at least one chain (with the **−segid** flag) for which the calculation will be performed. Because carma's default is to use only the CA atoms, you most probably also need a **−atmid HEAVY** or **−atmid ALLID** flag.

−DISTANCE [<ATOM1> <ATOM2>] | [FILE NAME]

The **−distance** flag is followed by two integers which correspond to the indeces of two atoms whose distance the program will calculate (as a function of frame number). If you want to calculate many distances simultaneously, you can feed a file containing the atomic indeces. The atomic indeces are one−based, and if you use the **−atmid ALLID** flag to select all atoms, then the numbers you want are the atom numbers as contained in the PSF file. To make this clear: if you run the program by saying

```
carma -v -dist 1 2 my.dcd my.psf
```

then, **carma** will calculate the distance between the first **CA atom** and the second **CA** atom. The reason is that the program's default is to only use the CA atoms. The way to avoid complications is to say

```
carma -v -atmid ALLID -dist 13 28 my.dcd my.psf
```

which will calculate the distance between the 13th atom declared in the PSF file (whichever this atom may be), and the 28th. The results will be written in a file named **carma.distances** in the current directory.

−TORSION [<ATOM1> ... <ATOM4>] | [FILE NAME]
The **−torsion** flag instructs the program to calculate the torsion angle between the four atoms whose indeces follow the card. If you want to calculate many torsions simultaneously, you can feed a file containing tetraplets of atom indeces. Please see the description of the **−distance** flag (above) for a clarification concerning atom selections. The results will be written in a file named **carma.torsions** in the current directory. With the wide −spread confusion between dihedral and torsion angles, you are better −off confirming that the numbers you get from **carma** indeed correspond to what you want to calculate.

−BEND <ATOM1> ... <ATOM3>
The **−bend** flag instructs the program to calculate the bending angle between the three atoms whose indices follow the card. Again, see the description of the **−distance** flag (above) for a clarification concerning atom selections. The results will be written in a file named **carma.bendangles** in the current directory.

−SURFACE
When this flag is present, the program will calculate a metric related to the total surface area of the atoms selected. To avoid misunderstandings and flames:

This is NOT an accessible surface area,
NOR a properly calculated total molecular surface area .

Now that this is clear, let me describe in some detail what carma actually calculates: Start by preparing a grid enclosing the molecule of interest using a 0.4 Angstrom distance between successive grid points. Then, set all grid points that are less than 3.2 Angstrom from any atom to a value of 1, everything else to zero (ie. make a fat molecular mask). Count the number of grid points that have a value of 1 and at least one of their closest neighbors is a zero. This is the number of grid points that form the surface of the mask. Multiply this number with a constant and this is what you get in the file **carma.surface.dat**. The constant has been chosen so that the numbers you get are approximately on the same scale as the values of the surface area (in Angstrom squared). In summary, what you get is essentially the number of 'pixels' making −up a mask around your molecule. Although the numbers per se are neither ASA nor SA, I do hope that their **relative** long−range changes for the length of the trajectory are representative of what happens with the total surface area, especially if you use this flag to calculate differences (corresponding to buried surfaces). Needless to say that if you use this flag to follow changes in the buried surface area of an oligomer, you should not even think about converting them to DeltaG.

AVERAGE AND RMSD (FROM AVERAGE) DISTANCE MAPS

−RMS
In this mode **carma** will produce two postscript files : The first (of the form **my.dcd.averag.ps**) will contain a map of the average values of all CA −CA distances for all frames contained in the DCD file. The second file (of the form **my.dcd.rmsdev.ps**) will contain the corresponding root−mean−square deviations from the aforementioned averages. The contrast of the image contained in this second file is reversed (the higher the RMSD, the darker the image).

−MRMS <FLOAT>
This keyword allows you to adjust the maximum value of the RMSD map that will be plotted. All values greater than **mrms** will appear black in the final postscript image.

RUNNING STRIDE TO CALCULATE SECONDARY STRUCTURE ASSIGNMENTS

−STRIDE

If you are on the unix machine, and if the executable for stride is present and in your path (say 'stride' from the unix shell to check) you can ask carma to automatically produce a file containing the stride−derived secondary structure assignments. The way to do that is to use something like '−pdb −stride −atmid HEAVY −segid PROT' in which case carma instead of producing PDB files, will pass them to 'stride' for the assignments' calculation ('PROT' is the segid of your protein component). The results will be found in a file named **carma.stride.dat** in the current directory (one line per trajectory frame).

If you have VMD installed, then you already have stride, but with a different name and at a directory that most probably than not, is not in your path. You'll need something like 'sudo /bin/cp /usr/local/lib/vmd/stride_LINUX /usr/local/bin/stride' to make copy it (with a suitable name) in /usr/local/bin.

POSTSCRIPT THINGS

−COLOUR [15 X <FLOAT>]

If the **−colour** flag is present in the command line, then instead of the default linear grayscale postscript, **carma** will emit colour postscript. If you try it and end −up believing that the author is colour−blind, then read−on: It is possible to define your own colour gradient by adding exactly 15 (!) numbers (all between 0.0 and 1.0) after the flag. These 15 numbers are the Red−Green−Blue components of your colour gradient at five points: at your data's minimum, at one quarter, at half, at three −quarters, and at maximum. The program's default can be found in the source code (search for RED_1), and goes from dark blue (for small values), through yellow (for intermediate values), to dark red (for large values).

−MIN <FLOAT>

The linear grayscale (or colour) gradient of the CA −CA distance maps will start at the specified distance. All distances shorter than this will appear black in the postscript image. This keyword also defines the smallest visible value in the case of variance −covariance calculations and when using the program as a filter.

−MAX <FLOAT>

The linear grayscale (or colour) gradient of the CA −CA distance maps will stop at the specified distance. All distances longer than this will appear white in the postscript image. This keyword also defines the largest visible value in the case of variance −covariance calculations and when using the program as a filter.

−REVERSE

This flag will reverse the contrast of the postscript plots produced by the program.

PCA−BASED FREE ENERGY LANDSCAPES

The DeltaG plots are obtained from the equation DeltaG = −kB T ln[P/Pmax] where kB is Boltzmann's constant, T is the temperature in Kelvin, and P and Pmax are probabilities obtained from the distribution of the principal components for each structure (frame) from the trajectory. By dividing with Pmax, the minimum DeltaG is zero at the configuration corresponding to Pmax. To keep the colour gradient as linear as possible within the range of interest, **carma** assigns the value of Pmin to all zero−valued parts of the probability distribution, where Pmin is the lowest non−zero P observed.

If you want to convince **carma** to produce graphs with a fixed scale, use the keyword **−DGwidth <float>** where **<float>** is the length in Angstrom of the edge of the graphs (if, for example, you want all your DeltaG plots to range from −20 Angstrom to +20 Angstrom, then add the flag **−DGwidth 40**.

PCA−BASED CLUSTER ANALYSIS

If the flags **−proj** or **−dPCA** are used and their second argument is greater than 2, carma will automatically perform an analysis of the three largest principal components with the aim of

identifying prominent molecular configurations corresponding to heavily populated clusters in the three−dimensional principal component space (see below for a detailed description of what the program does). The results from this calculation are placed in a file name **carma.clusters.dat** whose structure will look similar to this:

```
  1   1   -0.9946293    0.5751649   -0.3420320
  3   1   -1.0408418    0.6531497   -0.2453037
 ........................................
785   3   -0.4106219   -0.6585305   -0.5737749
786   3   -0.3866465   -0.7197343   -0.3226163
 ........................................
```

The first column is the frame number from the DCD file, the second column is the cluster identifier. Cluster number 1 is the cluster with the highest density (not neccessarily the most populated cluster, see below), number 2 the one with the second highest density, etc. The rest of the columns are the values of the three largest principal components for the corresponding frames. The reason that these are included is to allow you to check visually what the program did by comparing the distribution from the file **carma.dPCA.fluctuations.dat** (or **carma.PCA.fluctuations.dat**) with the distributions obtained if only frames from a specific cluster are used (see the **I hate manuals** tutorial above).

What carma does: For each trajectory frame calculate the values V1, V2, V3 of the three largest principal components. Use these values to populate a three dimensional map describing the distribution of these values (the higher the value at a grid point of the map, the larger the number of frames with V1−V2−V3 values closest to this grid point). This map can be viewed with VMD if you pass carma the the **−3d** flag. At the next stage the program attempts to identify all isolated 'peaks' in this distribution map, corresponding to heavily populated clusters. The rmsd cutoff for identifying isolated peaks is determined automatically by examining the fraction of 'variance explained' as a function of the number of clusters (the file **carma.variance_explained.dat** contains the results from this calculation). The automatically −determined cutoff can be changed by passing an optional fourth argument to **−proj** or **−dPCA**. Now that the clusters have been identified, carma goes back and determines which trajectory frames belong to which cluster. The results are written to the file **carma.clusters.dat** described above.

Limitations: (1) If the distribution map is not sufficiently sampled, or if the degrees of freedom are too few, the map may not have the 'peaky' appearance assumed by the procedure described above (you can see whether this is the case by examining the DeltaG plots produced by the program). (2) The procedure does not even try to assign each and every frame to a cluster. Depending on the morphology of the distribution map, a significant percentage of trajectory frames will remain unassigned. For this reason the clustering procedure may not be useful for determining transitions and average lifetimes of conformations.

GRAPHICS TRICKS AND EXAMPLES

Carma can do a few primitive tricks with graphics. If you have a graphics −enabled executable, try giving

carma my.psf my.dcd

You should be asked for the number of the first frame to display. Just hit enter and you should see it.

While viewing the trajectory, use the keys 'x', 'y', and 'z' to select orthogonal view. Press the 'P' key to pause. Change the zoom with the '<' and '>' keys. Press 'S' to enter stereo (side−by−side) mode. Press the 'ESC' key to swing it (or to stop swinging it). Use the arrow keys to navigate. Press 'R' to reset translations. Press 'P' to exit pause mode (and continue viewing the trajectory). You can change the step (stride) by pressing the '+' and '−' keys.

You can use the **−segid** and **−atomid** flags to change which atoms are displayed (and how they

are coloured). See section on graphics keywords.

Note, that if you have an executable with graphics support, you can always switch −on the graphics (with something like **−gl 100 −trace**) irrespectively of what else you asked carma to do. Waiting for **carma** to finish will never be the same.

Since version 0.8, the program understands DCD files containing unit cell dimensions (for a PBC simulation). Additionally, it can produce a very primitive graphical representation of a volume containing 2x2x2 cells (press 'B' while viewing the trajectory).

In the examples that follow the number next to the ' −gl' flag corresponds to the size of the edge the graphics window in Angstrom.

carma my.dcd my.psf

carma −gl 80.0 −atmid ALLID solute.dcd solute.psf

carma −gl 80.0 −atmid ALLID −cpk solute.dcd solute.psf

carma −gl 80.0 −atmid ALLID −trace all_atom.dcd all_atom.psf

carma −gl 80.0 −atmid ALLID −cpk −trace all_atom.dcd all_atom.psf

carma −gl 80.0 −atmid ALLID −trace solute.dcd solute.psf

carma −gl 80.0 −atmid ALLID −cpk −trace solute.dcd solute.psf

carma −gl 80.0 −atmid ALLID −segid A −segid B −segid WT1 all_atom.dcd all_atom.psf

carma −gl 80.0 −atmid ALLID −segid A −segid B −sp −trace all_atom.dcd all_atom.psf

carma −gl 80.0 −atmid ALLID −segid A −segid B solute.dcd solute.psf

carma −gl 80.0 −trace −segid A −segid B solute.dcd solute.psf

carma −gl 80.0 −atmid N −atmid O −atmid CA −atmid C −cpk solute.dcd solute.psf

OTHER USES

USING CARMA AS A UNIX FILTER

If you have a square matrix of reals **carma** will happily read it in and write out a postscript file with the name **carma.stdin.ps** containing a grayscale image of your data. The way to do that is simply **carma − < myfile.dat**. If you also want to adjust the contrast of the image, give **carma −min ??? −max ??? − < myfile.dat**. If you also want your matrix formatted by carma, **carma −write −min ??? −max ??? − < myfile.dat** should do the trick.

MERGING POSTSCRIPT IMAGES

Executing **carma** with only arguments the names of two postscript files of same dimension (and produced by the program during previous runs) will give you a new postscript file with the name **carma.merged.ps** containing the (above the diagonal) half from the first image, and the other (below the diagonal) half from the second.

RE−ORDERING DCD FRAMES

Executing **carma** with the **−sort <my.order> my.dcd** arguments, will allow you to re−order the frames in the DCD file to match the order contained in the file **<my.order>**. The **my.order** file is an ASCII file which contains one frame number per line. The output DCD file will be **carma.reordered.dcd** and will contain the defined frames with the order given in the **<my.order>** file.

SHORT EXAMPLES

Read a PDB file and produce a postscript image of the CA −CA distances in the structure. Also produce a formatted ASCII file containing the actual distance matrix :

carma −write my.pdb

Read a DCD and its corresponding PSF file and produce postscript images of the CA −CA

distance maps for the trajectory frames 10,20,...,190 :

carma −first 10 −last 190 −step 10 my.psf my.dcd

Calculate the average and rmsd CA − CA distance map for all frames between 1000 and 4500. Be more talkative and do write out the actual matrices :

carma −verb −write −rms −first 1000 −last 4500 my.psf my.dcd

Calculate the average and rmsd CA − CA distance map for all frames between 1000 and 4500. Be more talkative and do write out the actual matrices. Also, change the contrast of both the average and the rmsd plots :

carma −verb −write −rms −max 10.0 −mrms 1.0 −first 1000 −last 4500 my.psf my.dcd

Calculate the atomic fluctuations cross −correlation matrix, write the actual numbers out, and only plot positive correlations :

carma −verb −write −cov −dot −min 0.0 my.psf my.dcd

Calculate the eigenvectors and eigenvalues of the Cartesian (x,y,z) fluctuations of the CA atoms' variance−covariance matrix :

carma −verb −write −cov −eigen my.psf my.dcd

Write a DCD file containing the CA motion due to the 10 largest eigenvalues :

carma −verb −write −cov −eigen −out 1 10 1 my.psf my.dcd

Write a DCD file containing a smooth depiction of the CA motion due to the largest eigenvalue−eigenvector pair and with a range of amplitudes from 15.0 to −14.5 :

carma −verb −write −cov −eigen −play 1 15.0 −14.5 my.psf my.dcd

A LONGER EXAMPLE

We start with four files : **my.dcd** and **my.psf** are from your MD run, **my_CAs.psf** is the PSF corresponding to the CA atoms only, **my_AB_CAs.psf** is the PSF corresponding to the CA atoms of the segments "A " and "B " of your structure (you can use VMD or X − PLOR to produce these).

Start by removing any global rotations/translations of the whole molecule (and considering only CAs) :

carma −verb −fit my.dcd my.psf

mv carma.fitted.dcd fitted_whole.dcd

mv carma.fit −rms.dat rms_whole.dat

Repeat the procedure, but this time using only segments A and B :

carma −verb −fit −segid " A " −segid " B " my.dcd my.psf

mv carma.fitted.dcd fitted_AB.dcd

mv carma.fit −rms.dat rms_AB.dat

You can use a plotting program to compare the evolution of the rmsds for the whole molecule vs the A−B chains (with something like **xmgr rms_whole.dat rms_AB.dat**).

Calculate the average CA − CA distance maps and their corresponding rms deviations for both (fitted) trajectories :

carma −verb −max 15.0 −mrms 2.0 my_CAs.psf fitted_whole.dcd

mv fitted_whole.dcd.averag.ps whole.averag.ps

mv fitted_whole.dcd.rmsdev.ps whole.rmsdev.ps

carma −verb −max 15.0 −mrms 2.0 my_AB_CAs.psf fitted_AB.dcd

mv fitted_AB.dcd.averag.ps AB.averag.ps

mv fitted_AB.dcd.rmsdev.ps AB.rmsdev.ps

gv whole.rmsdev.ps

gv AB.rmsdev.ps

Calculate the eigenvectors and eigenvalues of the Cartesian (x,y,z) fluctuations of the CA atoms' variance−covariance matrix but only considering the C −D segments (note that because we are not producing a DCD file, we do not need a PSF file just for segments C −D) :

carma −verb −segid " C " −segid " D " −write −cov −eigen my.psf my.dcd

Now, lets make a movie depicting the motion of the whole molecule (considering CAs only) due the eigenvector corresponding to the largest eigenvalue :

carma −verb −write −cov −eigen −proj 2 2 300 my.psf my.dcd

echo "Amplitude limits are " ‘sort −r −n −k 2 carma.fluctuations.dat | tail −1 | awk '{print $2}'‘ ‘sort −n −k 2 carma.fluctuations.dat | tail −1 | awk '{print $2}'‘

carma −verb −write −cov −eigen −play 1 <amp1> <amp2> my.psf my.dcd

vmd my_CAs.psf carma.play.dcd

where <**amp1**> and <**amp2**> are the values returned by the echo command line.

APPENDIX: PREPARATION OF CA −ONLY PSF FILES

Carma can only process "paired" DCD and PSF files: if a DCD file only contains CA atoms, you need a PSF file containing only CA atoms. If your DCD file contains only 'HEAVY' atoms, you need a PSF file containing only non −hydrogen atoms.

The easiest way to prepare such PSF files is through carma itself : when the ' −w' flag is being used, then carma will prepare a pseudo −PSF (suitable only for usage with carma) with the name **carma.selected_atoms.psf** which will contain information for the selected atoms. If, for example, you say 'carma −v −w −fit all_atoms.dcd all_atoms.psf' then carma will produce a CA −only DCD file plus a pseudo −PSF file which you can then rename to more suitable names :

```
mv carma.fitted.dcd CAs_fitted.dcd
mv carma.selected_atoms.psf CAs.psf
```

You can also prepare such PSF files using X −PLOR or VMD. If you are an X −PLOR user, the following script will suffice (protein.psf is the complete PSF file of your protein as produced, for example, from PSFGEN):

```
-----------------------------------------------

structure @protein.psf end
delete selection=( not (name ca)) end
write structure output=CAs.psf end
stop

-----------------------------------------------
```

If you'd rather use VMD, here is the script (protein.pdb and protein.psf are the PDB and PSF files as produced, for example, from PSFGEN):

```
-----------------------------------------------

mol load pdb protein.pdb
mol load psf protein.psf
set sel [atomselect 0 "backbone and name CA" ]
$sel writepsf CAs.psf
-----------------------------------------------
```

Slight modification of these scripts can allow you to prepare PSF files for various atom combinations, for example, all non−hydrogen protein atoms.

BUGS & FEATURES

* The documentation is hell.

* The documentation is a mess.

* **carma**'s sanity checks on input are rudimentary. It should be relatively easy to dump core with random combinations of keywords and parameters.

* The PDB files written by **carma** use only three characters for atom names and only one character for the chain identifier. For example, HG21, HG22, HG23 will all be written to the PDB file as HG2, and segids WT1, WT2, ..., will all take the chain identifier 'W'.

* **carma** is known to process correctly PSF and DCD files written by X−PLOR and NAMD, but may give **wrong** results with CHARMM DCD files if these contain additional CHARMM−specific blocks within the coordinate sets. **carma** does check for the presence of CHARMM−specific flags in the header block and if found it will abort the calculation. Still, better safe than sorry :

Using this program with CHARMM−type DCD files may under certain circumstances give you wrong results

* When **carma** calculates fluctuations for dihedral PCA, it subtracts the average sin/cos values. This is different from the original dPCA papers by the Stock group. This corresponds to a shift of origin for the PCs and does not affect the analysis. Thanks to Jerome Henin for pointing this out.

* **carma** takes a rather cavalier approach with its input files, just trying to get to the needed coordinates as quickly as possible ignoring most of the structure of these files. The result is that the program may not perform as expected in hypothetical cases where this additional structure ought to have been interpreted.

* **carma** is using single precision eigenroutines (whether LAPACK's or numerical recipes'). The implication is that the resulting eigenvalues and eigenvectors are of limited accuracy. To clarify this point, the following table shows the relative (per−cent) discrepancy

$$100 * \text{SUM\_i} \mid \text{L1(i)} - \text{L2(i)} \mid / \text{SUM\_i} \; [0.5*((\text{L1(i)}+\text{L2(i)})]$$

between the eigenvalues of a 551 by 551 Wilkinson matrix as calculated from executables produced by two different compilers and two different eigenroutine libraries on a linux machine :

Compilers	gcc_LA	gcc_NR	intel_LA	intel_NR
gcc_LA	−	0.000045	0.000040	0.000046
gcc_NR	0.000045	−	0.000045	0.000052
intel_LA	0.000040	0.000045	−	0.000047
intel_NR	0.000046	0.000052	0.000047	−

Since for a Wilkinson matrix the eigenvalues should be equal in pairs (to at least 12 digits), it is also possible to access the internal reliability of the results by calculating

$$100 * \text{SUM} \mid \text{L(i)} - \text{L(i+1)} \mid / \text{SUM} \; [0.5*(\text{L(i)}+\text{L(i+1)})]$$

where the sums are taken (in pairs) over the 200 largest eigenvalues :

gcc_LA	gcc_NR	intel_LA	intel_NR
0.000039	0.000102	0.000078	0.000092

The different behaviour of the LAPACK routines with the two compilers is probably due to the greedy optimisation flags that I have used while making the LAPACK library with the Intel compilers.

VERSION

Version: 1.7 of March 2017

AUTHOR

Nicholas M. Glykos, Dept of Molecular Biology and Genetics, Democritus University of Thrace, Greece. Please send comments, suggestions, flames and bug reports to **glykos@mbg.duth.gr**

LATEST VERSION

You can get the latest release of the program via **http://utopia.duth.gr/glykos/**