Introduction

Java Essential Dynamics (JED) is a set of java programs for analyzing protein trajectories. The trajectories may be derived from MD, FIRST/FRODA, or any other dynamic simulations that output a trajectory as a set of PDB files. The program can handle single chain PDB files with no chain identifier as well as multi chain PDB files that use chain IDs. The user may specify the set of residues to be considered for the analysis, and this set need not be continuous. As a pre-processing step, JED reads in all PDB files in a specified directory and aligns all the structures in the trajectory to a specified reference structure using a quaternion alignment algorithm. A matrix of the read **PDB coordinates** obtained from all the residues in the PDB files is created that can be used for all additional JED runs. Additionally, a **list of all the residues** found in the PDB files (along with the chain IDs when appropriate) is generated. The original and transformed **conformation rmsd** is determined for each trajectory structure using a specified reference structure and the **residue rmsd (RMSF)** is determined from the entire trajectory. An **edited PDB file** is generated in which the B-factors have been replaced with the residue rmsd values for visualization purposes.

The core element of essential dynamics is performing principle component analysis (PCA) and JED implements two variations of PCA. The first and most common method is a Cartesian coordinate based analysis (**cPCA**)using the selected set of alpha carbons. The second method is an internal coordinate based analysis using the distances (**dPCA**) between each alpha carbon and all the others in the protein. This is a computationally expensive algorithm and the interpretation is difficult for sets of residues greater than 10 (note that dPCA on a ten residue subset will yield eigenvectors having $n*(n-1)/2 = 45$ components, each corresponding to one set of inter-residue distances). The user can specify the number of top Cartesian and top Distance modes to be determined, as well as the number of Cartesian modes to visualize. Mode visualization is done by creating a set of 20 PDB files that capture the displacement of the alpha carbons for the given mode. A scale-factor parameter is adjustable to control the amount of displacement in the modes. A Pymol© script is generated to animate the frames.

The two variants of PCA are implemented using both the covariance matrix (**Q**) and the correlation matrix (**R**), as these two methods yield somewhat different results due to the inherent statistical biases in each approach. JED handles the removal of outliers prior to the PCA analyses with two approaches. First, the user can specify the **percent** of the structures to be removed based on the conformation rmsd. Thus, the most deviant or under-sampled structures can be removed prior to the PCA analysis. Second, the user can specify a **z-score cutoff** such that any coordinate value that exceeds the |cutoff| is replaced with the mean for that coordinate. This type of conditioning has the effect of greatly reducing the condition numbers of Q and R. Both approaches are optional and can be turned off by setting the associated parameters to zero. JED computes the **PCA modes** (RMSD and MSD, with and without weight by the corresponding eigenvalue) from the Cartesian eigenvectors so that they may be mapped to the residue set. As noted above, sets of structures can be generated to visually inspect the cPCA modes. Eigenvectors from the distance PCA cannot be mapped to the residue set in any simple way, so no mapping or visualization is attempted.

A set of **displacement vectors** (DV) is calculated using a specified reference structure and then those DVs are projected onto a set of eigenvectors creating the delta vector projections (DVPs), also known as the principle components (**PCs**), which allow the user to investigate how the trajectory projects into the space defined by each eigenvector. The **PCs** are given using un-normalized and normalized inner products, as well as weighted by the corresponding eigenvalue.

JED performs a simple subspace analysis (**SSA**) on the two equidimensional sets of eigenvectors generated from the Q and R variants of PCA. This allows one to quantitatively determine how different the PCA results are due only to the choice of PCA model. A comparison is also made to a random set of eigenvectors derived from the same vector space. This allows the user to establish a baseline for statistically valid subspace comparisons. Additional analysis can be done using the Subspace Analysis classes or the associated driver programs. To perform these kinds of tests, one first needs to generate sets of eigenvectors from each trajectory of interest as well as from the pooled trajectories to use as a reference set. Subspace analysis is done by comparing the sets of eigenvectors and determining the **RMSIP**, Principal Angles (**PAs**), cumulative overlap (**CO**), cosine products (**CP**), vectorial angular sum (**VAS**), and the maximum angle between subspaces of the given vector space.

```
(In this tutorial, code, file paths, and text file content are shown in dark blue 9 point Consolas)
```

Understanding the Program

JED is a Java based set of programs and as such can run on any platform with a suitable Java Runtime Environment (JRE).

The program can be run from compiled source or from the provided executable jar files. It is critical that the environment variable Java **CLASSPATH** be correctly set. Alternatively, you can always specify the complete absolute path for files and and/or add the -cp option to the java command.

There are two driver programs for JED: One (**JED_Driver**) runs a single job using parameters specified in the input file, and the other (**JED_Batch_Driver**) runs a batch of jobs sequentially. The first is suited for running a single job at the command line or when using submit scripts on computer cluster resources. This can be implemented using job arrays so that your jobs run in parallel rather than sequentially. The second is suited for running multiple jobs on a single computer while maximizing available compute resources.

Since JED requires an input file for job parameters, the run command takes one argument: the path to the input file. If no argument is specified, then JED assumes that the default input file named is used and the file is located in the same directory from which the JVM was called. The default input file names are:
JED_Driver.txt for JED_Driver.java (or jar file)
JED_Batch_Driver.txt for JED_Batch_Driver.java (or jar file)

To run **JED_Driver** at a command prompt or in a PBS script, you can use one of the following commands:

```
Java -d 64 JED_Driver "/path/to/your/input/file.txt" (runs the compiled java program)
Java -jar -d 64 JED_Driver.jar "/path/to/your/input/file.txt" (runs the executable jar file)
```

To run **JED_Batch_Driver** at a command prompt or in a PBS script, you can use one of the following commands:

```
Java -d 64 JED_Batch_Driver "/path/to/your/input/file.txt")
Java -jar -d 64 JED_Batch_Driver.jar "/path/to/your/input/file.txt"
```

Note: Most computer clusters have a 64 bit JRE installed. Be sure to request adequate memory resources as JED computes the complete covariance matrix and holds it in memory. Additionally, matrix diagonalization scales as (O_3) so depending on the size of your protein, you may need 12-36GB . For very large jobs (thousands of residues and tens of thousands of frames), it is advisable to use a large memory node (1TB).

Each job should be assigned to its own directory, which should contain either the PDB files to read or the coordinate file to process, along with the reference PDB file and residue lists for specifying the Cartesian and Distance subsets.

Output files are written to sub-directories of the working directory, structured to organize the multitude of files produced in a meaningful manner. The top level of this directory tree is named "JED_Results_**$description**", where **$description** is a user set parameter that succinctly describes the job. Limbs of the tree separate cPCA, dPCA, and mode visualization analyses (VIZ), when present. Each of these in turn contains limbs for Q and R compartmentalization. Most of the output file names include the **number of residues** in the selected subset for reference plus a description of the file contents.

Limitations: The PDB files are initially read to obtain the x, y, and z coordinates of the alpha carbon positions and create a matrix of these coordinates for essential dynamic analysis. **Each PDB file must have the exact same number of residues to be processed by JED**. While JED can handle a variety of numbering schemes by mapping the residues present in the PDB files to the coordinate matrix, it can NOT process **occupancy** values. **If an MD simulation outputs PDB files with multiple coordinate positions and occupancy values, these CAN NOT BE USED for JED.** Finally, some thought should be given to the naming of the PDB files as this effects how the files are sorted, which determines the order of the frames in the coordinate matrix. **Failure to pad a numbered list of files with sufficient leading zeros will**

lead to orderings such as 1,10,100,1000 rather than 1,2,3,4. It is therefore strongly suggested to ensure proper naming to avoid these situations.

Running the Program

IN ALL CASES, A preliminary run with NO PCA must be performed to generate the JED formatted coordinate matrix file for all the alpha carbons in the PDB files. This makes multiple subset analyses much more efficient as the program only needs to read in the PDB files once. It also serves to guarantee that the specified residues for subset selection are correctly chosen. After this step, the PDB files can either be deleted or archived, with the exception of the reference PDB file. Once the coordinate matrix is created, it can be used for all future analyses using different residue subsets and different job parameters.

The name of the coordinate file matrix produced from the PDB files is: `"original_PDB_coordinates.txt"`
The matrix packing is as follows:
Rows are coordinate variables and columns are frames.
For N residues, there are 3N rows: N x-coordinates, N y-coordinates, and N-z coordinates, stacked in that order.

The file to use in all subsequent JED analyses is the original_PDB_coordinates matrix.

Note: The most critical step in preparing to run JED is in the creation of the input file. The input file must have the correct format (shown in examples below) and the entries must be accurate. If either of these conditions is violated, the program will crash, or worse, the results will be corrupt.

Notice!
If ANY directory cannot be found or ANY file cannot be stat, the program crashes.
If there is any problem with any input, the program crashes.
It was coded to do exactly this, so as to prevent garbage results.

The Preliminary Run

run command:

```
java -jar -d 64 JED_Driver.jar "/path/JED_Driver.txt"
```

The PDB files (including the PDB reference file) must be in the working directory.
JED_Driver.txt may be in the working directory.

This pre-processing step will read all PDB files in the working directory, but will perform NO PCA.
The purpose of this is to generate the matrix of coordinates for performing subset analyses efficiently.

Standard NO-PCA Output Files:
These are written to the **root** of the JED Results directory tree:
`/working/directory/JED_Results_Description/`

JED LOG providing a summary of the job parameters and results:

`JED_Log.txt`

PDB READ LOG listing all the PDB files read, in the order they were read:

`PDB_READ_Log.txt`

coordinates matrix from all the alpha carbon coordinates in the PDB files:

`original_PDB_coordinates.txt`

transformed coordinates matrix, which aligns all the frames to the reference frame :

`ss_$num_res_transformed_PDB_coordinates.txt`

list of all residues found in the PDB files for subsequent editing and use:

`All_PDB_Residues_JED.txt`

`All_PDB_Residues_Multi_JED.txt (for Multi runs)`

original and transformed conformation RMSDs:

`ss_$num_res_original_conformation_rmsds.txt`

`ss_$num_res_conformation_rmsds.txt`

residue RMSDs (RMSF):

`ss_$num_res_residue_rmsd.txt`

edited PDB file containing all the residues with the RMSF replacing B-factors:

`ss_$num_res_edidited.PDB`

coordinates Z-Score matrix from all the alpha carbon coordinates in the PDB files:

`ss_$num_res_coordinate_Z_scores.txt`

percent of the frames to remove based on conformation RMSD (OPTIONAL)

z-cutoff for adjusting coordinate outliers to their mean values (OPTIONAL)

If these parameters are set to a value other than zero, additional output files are generated with the results.

However, the appropriate time to handle outliers is during runs that actually perform PCA and it is recommended that for pre-processing runs, these be set to zero.

Below is a sample input file for Single Chain PDB files:

```
-------------------------------------------------------------------------------------------------
0.00
0.00
1     0
0     0     0
/working/directory/job/
Description     reference_PDB_file.pdb
-------------------------------------------------------------------------------------------------
```

Notes: This is a tab (space) separated file. There must be NO LEADING blank lines or spaces (no trailing too).

Line 1 specifies the **percent** (double) of frames to remove from the data: 0 →NONE

Line 2 specifies the **z-cutoff** (double) for adjusting outliers: 0 →NO ADJUSTMENT

Line 3 Token 1 specifies whether to **read PDB files** (0 or 1) →1 = yes

Line 3 Token 2 specifies if the PDB files are **Multi** Chain (0 or 1) → 0 = no

Line 4 Token 1 specifies the **number of Cartesian** (integer) modes to process →0 = none

Line 4 Token 2 specifies the **number of Distance** (integer) modes to process →0 = none

Line 4 Token 3 specifies the **number of Cartesian modes to Visualize**(integer) →0 = none

Line 5 specifies the **working directory** (String)

Line 6 Token 1 specifies the job **description** (String) for job one

Line 6 Token 2 specifies the **reference PDB** (String) for job one

----

Warning:
Both number of modes Cartesian and number of modes Distance MUST be set to ZERO.
This tells JED to perform the NO-PCA analysis.

The Read flag MUST be set to ONE and the MULTI flag MUST be set to ZERO.

All PDB files MUST have the same number of residues. The matrix of alpha carbon coordinates is determined from the first PDB file read. If other files in the working directory do not match exactly, then the array sizes will not match and the program will crash. IF JED crashes during the reading of PDBs, this is probably the reason.

In subsequent analyses, it is critical that no residues are requested that do not actually exist in the PDB files!
JED maps the specified residue list to an internal list that is aligned to the columns of the coordinates matrix.

The file to use in all subsequent JED analyses is the original_PDB_coordinates matrix.

This matrix contains all the residues in the PDB files and thus can be used for any subset of those residues. When a subset is chosen, a new correspondence set is generated and a new transformation is done to optimize the alignment of the structures. This removes overall translation and rotation.

Below is a sample input file for Multi-Chain PDB files:

Differences from Single-Chain analysis shown highlighted in amber

```
----------------------------------------------------------------------------------------
0.00
0.00
1     1     2     A     B     795     151     0     0
0     0     0
/working/directory/job/
Description     reference_PDB_file.pdb
----------------------------------------------------------------------------------------
```

Notes: This is a space delineated file. There must be NO LEADING blank lines or spaces (no trailing too).

Line 1 specifies the **percent** (double) of frames to remove from the data: 0 →NONE
Line 2 specifies the **z-cutoff** (double) for adjusting outliers: 0 →NO ADJUSTMENT
Line 3 Token 1 specifies whether to **read PDB files** (0 or 1) →1 = yes
Line 3 Token 2 specifies if the PDB files are **Multi** Chain (0 or 1) → 1 = yes
Line 3 Token 3 specifies the number of chains → (2);
Line 3 Token 4 specifies the first chain ID → (A);
Line 3 Token 5 specifies the second chain ID → (B);
Line 3 Token 6 specifies the number of residues in chain A→ (795);
Line 3 Token 7 specifies the number of residues in chain B→ (151);
Line 3 Token 8 specifies the offset of Chain A → (0);
Line 3 Token 9 specifies the offset of Chain B → (0);
Line 4 Token 1 specifies the **number of Cartesian** (integer) modes to process →0 = none
Line 4 Token 2 specifies the **number of Distance** (integer) modes to process →0 = none
Line 4 Token 3 specifies the **number of Cartesian modes to Visualize**(integer) →0 = none
Line 5 specifies the **working directory** (String)
Line 6 Token 1 specifies the job **description** (String) for job one
Line 6 Token 2 specifies the **reference PDB** (String) for job one

----

Warning:
Both number of modes Cartesian and number of modes Distance MUST be set to ZERO.
This tells JED to perform the NO-PCA analysis.

The Read flag MUST be set to ONE and the MULTI flag MUST be set to ONE.

All PDB files MUST have the same number of residues. The matrix of alpha carbon coordinates is determined from the first PDB file read. If other files in the working directory do not match exactly, then the array sizes will not match and the program will crash. IF JED crashes during the reading of PDBs, this is probably the reason.

Additionally, Multi Chain PDBs MUST have unique chain identifiers for every chain present in the file.
A missing chain identifier will cause JED to crash.

In subsequent analyses, it is critical that no residues are requested that do not actually exist in the PDB files!
JED maps the specified residue list to an internal list that is aligned to the columns of the coordinates matrix.

The file to use in all subsequent JED analyses is the original_PDB_coordinates matrix.

This matrix contains all the residues in the PDB files and thus can be used for any subset of those residues. When a subset is chosen, a new correspondence set is generated and a new transformation is done to optimize the alignment of the structures. This removes overall translation and rotation.

Debugging Crashes Part I:

Things that will generally make your life miserable…

Dumb mistakes:
Does the directory exist? If not, → CRASH!!!
Does the path to the input file exist? If not, → CRASH!!!
Does the input file exist in the proper location? If not, → CRASH!!!
Does the input-file start on the first line? If not, → CRASH!!!
The directory contains PDB files of different sizes → CRASH!!!
The directory does not contain the reference PDB file → CRASH!!!
Is that integer really an integer or is it a float or a string? → CRASH!!!
Does the directory string end in "/" or "\\"?: If not → CRASH!!!
Are the 0 and 1 flags set correctly? If not → CRASH!!!

More subtle mistakes:
The directory contains PDB files with 2 chains, but no chain IDs → Non-sense results!
If the PDB file numbering starts at 5, then the chain offset is not 0, it is 4 → Garbage out!

If the PDB file names are sorted in a different order than how they were generated, then the conformation RMSD results will not reflect what actually occurred in the simulation. This can be prevented by naming the PDB files appropriately, specifically, padding the numbers with sufficient leading zeros to ensure proper sorting. Note that this is not just a Java thing, it is a Unix thing too.

If the conformation RMSD is very different from what you expect, then you may be using PDB files that contain occupancy information. JED does not use that information. Your results will not be accurate.

If you have pooled data, make sure the combined matrix is accurate.

Performing Cartesian PCA

run command:

```
java -jar -d 64 JED_Driver.jar “/path/JED_Driver.txt”
```

The working directory must contain: The coordinates matrix, the PDB reference file, and the residue list.
It may also contain JED_Driver.txt

The purpose of this type of run is to perform Essential Dynamics using cPCA based on Q and R.
The user specifies the subset of interest for the analysis, which may be the entire protein or a small, non-contiguous selection of residues, by providing a residue list file. This task is simplified since JED has already created a list of all the residues in the protein. The user must simply edit this file. The cPCA results are written to the sub-directory "cPCA" and the Visualizations of the top modes (when selected) are written to the subdirectory "viz". The directory cPCA has sub directories for the Q and R analysis, as does the viz directory.

Standard Output Files:
These are written to the **root** of the JED Results directory tree:
`/working/directory/JED_Results_Description/`

JED LOG providing a summary of the job parameters and results:

`JED_Log.txt`

subset transformed coordinates matrix:

`ss_$num_res_transformed_PDB_coordinates.txt`

original and transformed conformation RMSDs:

`ss_$num_res_original_conformation_rmsds.txt`

`ss_$num_res_conformation_rmsds.txt`

residue RMSDs (RMSF):

`ss_$num_res_residue_rmsd.txt`

subset edited PDB file with the RMSF replacing B-factors:

`ss_$num_res_edidited.PDB`

subset coordinates Z-Score matrix:

`ss_$num_res_coordinate_Z_scores.txt`

list of frames removed, based on the **percent** parameter:

`ss_$num_res_ Removed_Conformation_Outliers.txt`

trimmed transformed coordinate matrix:

`ss_$num_res_ trimmed_$percent_percent_PDB_coordinates_COLS.txt`

list of coordinate variables adjusted, based on the **z-cutoff** parameter:

`ss_$num_res_ adjustments_per_variable.txt`

adjusted transformed coordinate matrix:

`ss_$Z_threshold_$z-cutoff_adjusted_PDB_coordinates_ROWS.txt`

<u>Standard cPCA Output Files:</u>

These are written to the **/cPCA subdirectory** of the JED Results directory tree:
`/working/directory/JED_Results_Description/cPCA/`

centroids (means) of the variables:

`ss_$num_res_centroids_of_variables.txt`

standard deviations of the variables:

`ss_$num_res_std_devs_of_centered_variables.txt`

displacement vectors:

`ss_$num_res_delta_vectors.txt`

<u>The COV output files are written to the /COV subdirectory of /cPCA</u>

`/working/directory/JED_Results_Description/cPCA/COV/`

<u>The CORR output files are written to the /CORR subdirectory of /cPCA</u>

`/working/directory/JED_Results_Description/cPCA/CORR/`

covariance matrix:

`ss_$num_res_COV_matrix.txt`

eigenvalues:

`ss_$num_res_eigenvalues_COV.txt`

top eigenvalues:

`ss_$num_res_top_$num_of_cart_modes_eigenvalues_COV.txt`

top eigenvectors:

`ss_$num_res_top_$num_of_cart_modes_eigenvectors_COV.txt`

top pca modes and top weighted pca modes:

`ss_$num_res_top_$num_of_cart_modes_pca_modes_COV.txt`

`ss_$num_res_top_$num_of_cart_modes_weighted_pca_modes_COV.txt`

top square pca modes and top weighted square pca modes:

`ss_$num_res_top_$num_of_cart_modes_square_pca_modes_COV.txt`

`ss_$num_res_top_$num_of_cart_modes_weighted_square_pca_modes_COV.txt`

top PCs and top weighted PCs:

`ss_$num_res_top_$num_of_cart_modes_PCs_COV.txt`

`ss_$num_res_top_$num_of_cart_modes_weighted_PCs_COV.txt`

top PCs and top weighted PCs:

`ss_$num_res_top_$num_of_cart_modes_normed_PCs_COV.txt`

`ss_$num_res_top_$num_of_cart_modes_weighted_normed_PCs_COV.txt`

Standard SSA Output Files:

These are written to the **/SSA subdirectory** of /cPCA:
`/working/directory/JED_Results_Description/cPCA/SSA/`

The Fast SSA Iterated Log:

`JED_FSSA_Iterated_log.txt`

The SSA Log:

`JED_SSA_dim_$top_num_cart_modes_log.txt`

The Random SSA Log

`JED_Random_SSA_log.txt`

There are many more files in the /SSA directory that are flat files of the results reported in the log files:

RMSIPs

PAs

COs

Cosine Products

Vectorial Sum of Angles

Below is a sample input file for Single-Chain PDB files:

```
-----------------------------------------------------------------------------------
0.01
3.00
0     0
20    0     2     1.0
/working/directory/
Description     reference_PDB_file.pdb
residues.txt
original_PDB_Coordinates.txt        0
-----------------------------------------------------------------------------------
```

Notes:

Line 1 specifies the **percent** (double) of frames to remove from the data: 0 → 1%

Line 2 specifies the **z-cutoff** (double) for adjusting outliers: 0 → values beyond |3.0|

Line 3 Token 1 specifies whether to **read PDB files** (0 or 1) → 0 = no

Line 3 Token 2 specifies if the PDB files are **Multi** Chain (0 or 1) → 0 = no

Line 4 Token 1 specifies the **number of Cartesian** (integer) modes to process → 20

Line 4 Token 2 specifies the **number of Distance** (integer) modes to process → 0 = none

Line 4 Token 3 specifies the **number of Cartesian modes to Visualize**(integer) → 2 = top two modes

Line 4 Token 4 specifies the **scale factor** for the visualizations (double) → 1.0 = default value

Line 5 specifies the **working directory** (String)

Line 6 Token 1 specifies the job **description** (String)

Line 6 Token 2 specifies the **reference PDB** (String)

Line 7 specifies the **residue list** (String)

Line 8 Token 1 specifies the **coordinate matrix** (String)

Line 8 Token 2 specifies the **reference column** (integer)

Warning:

Both the Read PDB file flag and the Multi flag MUST be set to ZERO.
This tells JED to perform the analysis on Single Chain PDBs.

The number of cPCA modes MUST not be zero.

There must be a Cartesian residue list specified.

The number of dPCA modes MUST be zero.

There must NOT be a Distance residue list specified.

If the subset you have chosen contains N residues, then you must NOT request more than 3N modes.

If you request N cPCA modes then you can only visualize up to N top modes.

If number of cPCA modes > 0, then there must be a Cartesian residue list file!

Below is a sample input file for Multi-Chain PDB files:

Differences from Single-Chain analysis shown highlighted in amber

```
-----------------------------------------------------------------------------------------
0.01
3.00
0     1     2     A     B     795     151     0     0
20    0     2     1.0
/working/directory/
Description     reference_PDB_file.pdb
residues.txt
original_PDB_Coordinates.txt        0
-----------------------------------------------------------------------------------------
```

Notes:

Line 1 specifies the **percent** (double) of frames to remove from the data: 0 → 1%

Line 2 specifies the **z-cutoff** (double) for adjusting outliers: 0 → values beyond |3.0|

Line 3 Token 1 specifies whether to **read PDB files** (0 or 1) → 0 = no

Line 3 Token 2 specifies if the PDB files are **Multi** Chain (0 or 1) → 1 = yes

Line 3 Token 3 specifies the number of chains → (2);

Line 3 Token 4 specifies the first chain ID → (A);

Line 3 Token 5 specifies the second chain ID → (B);

Line 3 Token 6 specifies the number of residues in chain A→ (795);

Line 3 Token 7 specifies the number of residues in chain B→ (151);

Line 3 Token 8 specifies the offset of Chain A → (0);

Line 3 Token 9 specifies the offset of Chain B → (0);

Line 4 Token 1 specifies the **number of Cartesian** (integer) modes to process → 20

Line 4 Token 2 specifies the **number of Distance** (integer) modes to process → 0 = none

Line 4 Token 3 specifies the **number of Cartesian modes to Visualize**(integer) → 2 = top two modes

Line 4 Token 4 specifies the **scale factor** for the visualizations (double) → 1.0 = default value

Line 5 specifies the **working directory** (String)

Line 6 Token 1 specifies the job **description** (String)

Line 6 Token 2 specifies the **reference PDB** (String)

Line 7 specifies the **residue list** (String)

Line 8 Token 1 specifies the **coordinate matrix** (String)

Line 8 Token 2 specifies the **reference column** (integer)

Warning:

The read PDB file flag MUST be set to ZERO.

The Multi flag MUST be set to ONE

This tells JED to perform the analysis on Multi Chain PDBs.

The number of cPCA modes MUST be **>** zero.

There must be a Cartesian residue list specified.

The number of dPCA modes MUST be zero.

There must NOT be a Distance residue list specified.

If the subset you have chosen contains N residues, then you must NOT request more than 3N modes.

If you request ***N*** cPCA modes then you can only visualize up to ***N*** top modes.

****

The format of the residue list file is TWO columns for Multi Chain Analysis: Chain ID, residue number.

The format of the residue list file is ONE column for Single Chain Analysis: residue number.

****

If number of cPCA modes > 0, then there must be a Cartesian residue list file!

Performing Distance PCA

run command:

```
java -jar -d 64 JED_Driver.jar "/path/JED_Driver.txt"
```

The working directory must contain: The coordinates matrix, the PDB reference file, and the residue list.
It may also contain JED_Driver.txt

The purpose of this type of run is to perform Essential Dynamics using dPCA based on Q and R.
The user specifies the subset of interest for the analysis, which is typically less than 10 residues, by providing a residue list file. This task is simplified since JED has already created a list of all the residues in the protein. The user must simply edit this file. The dPCA results are written to the sub-directory "dPCA". Note that for dPCA no transform is needed since internal distances are used for coordinates and no visualization can be done in JED for the distance modes. The directory dPCA has sub directories for the Q and R analysis as well as for the subspace analysis (SSA).

Note that choosing more than ten residues for the dPCA analysis makes the interpretation of the results quite challenging as each component of the distance eigenvectors corresponds to an inter-residue distance pair. Often subsets with three or four residues can be used to investigate experimental findings in critical areas like binding pockets or clefts.

Standard Output Files:

These are written to the **root** of the JED Results directory tree:
`/working/directory/JED_Results_Description/`

JED LOG providing a summary of the job parameters and results:

`JED_Log.txt`

subset transformed coordinates matrix:

`ss_$num_res_transformed_PDB_coordinates.txt`

original and transformed conformation RMSDs:

`ss_$num_res_original_conformation_rmsds.txt`

`ss_$num_res_conformation_rmsds.txt`

residue RMSDs (RMSF):

`ss_$num_res_residue_rmsd.txt`

subset edited PDB file with the <u>RMSF replacing B-factors</u>:

`ss_$num_res_edidited.PDB`

subset coordinates Z-Score matrix:

`ss_$num_res_coordinate_Z_scores.txt`

list of frames removed, based on the **percent** parameter:

`ss_$num_res_ Removed_Conformation_Outliers.txt`

trimmed transformed coordinate matrix:

`ss_$num_res_ trimmed_$percent_percent_PDB_coordinates_COLS.txt`

list of coordinate variables adjusted, based on the **z-cutoff** parameter:

`ss_$num_res_ adjustments_per_variable.txt`

adjusted transformed coordinate matrix:

`ss_$Z_threshold_$z-cutoff_adjusted_PDB_coordinates_ROWS.txt`

__**Standard dPCA Output Files:**__

These are written to the **/dPCA subdirectory** of the JED Results directory tree:

`/working/directory/JED_Results_Description/dPCA/`

subset distance residue stats from all the alpha carbon coordinates in the **subset**:

`ss_$num_res_distance_residue_stats.txt`

subset distance Z-Score matrix from all the alpha carbon coordinates in the **subset**:

`ss_$num_res_distance_Z_scores.txt`

subset all-to-all distances matrix from all the alpha carbon coordinates in the **subset**:

`ss_$num_res_all_to_all_distances.txt`

subset distance variables adjusted:

`ss_$num_res_ outliers_per_variable.txt`

subset centroids (means) of the variables:

`ss_$num_res_centroids_of_variables.txt`

subset standard deviations of the variables:

`ss_$num_res_std_devs_of_centered_variables.txt`

subset displacement vectors:

`ss_$num_res_delta_vectors.txt`

__**The COV output files are written to the /COV subdirectory of /dPCA**__

`/working/directory/JED_Results_Description/cPCA/COV/`

__**The CORR output files are written to the /CORR subdirectory of /dPCA**__

covariance matrix:

`ss_$num_res_distance_COV_matrix.txt`

eigenvalues:

`ss_$num_res_distance_eigenvalues_COV.txt`

top eigenvalues:

`ss_$num_res_top_$num_of_dist_modes_distance_eigenvalues_COV.txt`

top eigenvectors:

`ss_$num_res_top_$num_of_dist_modes_distance_eigenvectors_COV.txt`

top pca modes and top weighted pca modes:

`ss_$num_res_top_$num_of_dist_modes_distance_pca_modes_COV.txt`

`ss_$num_res_top_$num_of_dist_modes_weighted_distance_pca_modes_COV.txt`

top square pca modes and top weighted square pca modes:

`ss_$num_res_top_$num_of_dist_modes_square_distance_pca_modes_COV.txt`

`ss_$num_res_top_$num_of_dist_modes_weighted_square_distance_pca_modes_COV.txt`

top PCs and top weighted PCs:

`ss_$num_res_top_$num_of_dist_modes_PCs_COV.txt`

`ss_$num_res_top_$num_of_dist_modes_weighted_PCs_COV.txt`

top normed PCs and top weighted normed PCs:

```
ss_$num_res_top_$num_of_dist_modes_normed_PCs_COV.txt
ss_$num_res_top_$num_of_dist_modes_weighted_normed_PCs_COV.txt
```

Standard SSA Output Files:

These are written to the **/SSA subdirectory** of /cPCA:

```
/working/directory/JED_Results_Description/cPCA/SSA/
```

The Fast SSA Iterated Log:

```
JED_FSSA_Iterated_log.txt
```

The SSA Log:

```
JED_SSA_dim_$top_num_cart_modes_log.txt
```

The Random SSA Log

```
JED_Random_SSA_log.txt
```

There are many more files in the /SSA directory that are flat files of the results reported in the log files:

RMSIPs

PAs

COs

Cosine Products

Vectorial Sum of Angles

Below is a sample input file for Single-Chain PDB files:

```
-----------------------------------------------------------------------------------------------------
0.00
3.00
0     0
0     3     0
/working/directory/
Description     reference_PDB_file.pdb
residues_dist.txt
original_PDB_Coordinates.txt        0
-----------------------------------------------------------------------------------------------------
```

Notes:

Line 1 specifies the **percent** (double) of frames to remove from the data: 0 = none

Line 2 specifies the **z-cutoff** (double) for adjusting outliers: 0 → values beyond |3.0|

Line 3 Token 1 specifies whether to **read PDB files** (0 or 1) → 0 = no

Line 3 Token 2 specifies if the PDB files are **Multi** Chain (0 or 1) → 0 = no

Line 4 Token 1 specifies the **number of Cartesian** (integer) modes to process → 0 = none

Line 4 Token 2 specifies the **number of Distance** (integer) modes to process → 3 = top three modes

Line 4 Token 3 specifies the **number of Cartesian modes to Visualize**(integer) → 0 = none

Line 5 specifies the **working directory** (String)

Line 6 Token 1 specifies the job **description** (String)

Line 6 Token 2 specifies the **reference PDB** (String)

Line 7 specifies the **residue list** (String)

Line 8 Token 1 specifies the **coordinate matrix** (String)

Line 8 Token 2 specifies the **reference column** (integer)

Warning:

Both the read PDB file flag and the Multi flag MUST be set to ZERO.
This tells JED to perform the analysis on Single Chain PDBs.

The number of cPCA modes MUST be zero.

There must be no Cartesian residue list specified.

The number of dPCA modes MUST NOT be zero.

There must be a Distance residue list specified.

If the subset you have chosen contains ***N*** residues, then you must NOT request more than ***N(N-1)/2*** modes.

If number of dPCA modes > 0, then there must be a Distance residue list file!

Below is a sample input file for Multi-Chain PDB files:

Differences from Single-Chain analysis shown highlighted in amber

```
-------------------------------------------------------------------------------
0.00
3.00
0     1     2     A     B     795     151     0     0
0     3     0
/working/directory/
Description     reference_PDB_file.pdb
residues.txt
original_PDB_Coordinates.txt        0
-------------------------------------------------------------------------------
```

Notes:

Line 1 specifies the **percent** (double) of frames to remove from the data: 0 → none

Line 2 specifies the **z-cutoff** (double) for adjusting outliers: 0 → values beyond |3.0|

Line 3 Token 1 specifies whether to **read PDB files** (0 or 1) → 0 = no

Line 3 Token 2 specifies if the PDB files are **Multi** Chain (0 or 1) → 1 = yes

Line 3 Token 3 specifies the number of chains → (2);

Line 3 Token 4 specifies the first chain ID → (A);

Line 3 Token 5 specifies the second chain ID → (B);

Line 3 Token 6 specifies the number of residues in chain A→ (795);

Line 3 Token 7 specifies the number of residues in chain B→ (151);

Line 3 Token 8 specifies the offset of Chain A → (0);

Line 3 Token 9 specifies the offset of Chain B → (0);

Line 4 Token 1 specifies the **number of Cartesian** (integer) modes to process → 0

Line 4 Token 2 specifies the **number of Distance** (integer) modes to process → 3 = Top 3 modes

Line 4 Token 3 specifies the **number of Cartesian modes to Visualize**(integer) → 0

Line 5 specifies the **working directory** (String)

Line 6 Token 1 specifies the job **description** (String)

Line 6 Token 2 specifies the **reference PDB** (String)

Line 7 specifies the **residue list** (String)

Line 8 Token 1 specifies the **coordinate matrix** (String)

Line 8 Token 2 specifies the **reference column** (integer)

Warning:
The read PDB file flag MUST be set to ZERO.
The multi flag must be set to ONE;
This tells JED to perform the analysis on MULTI Chain PDBs.

The number of cPCA modes MUST be zero.
There must be no Cartesian residue list specified.
The number of dPCA modes MUST NOT be zero.
There must be a Distance residue list specified.
If the subset you have chosen contains ***N*** residues, then you must NOT request more than ***N(N-1)/2*** modes.

Remember that the format of the residue list file is TWO columns for Multi Chain Analysis: **Chain ID, residue number** while for Single Chain Analysis, the file has ONE column: **residue number**.

If number of dPCA modes > 0, then there must be a Distance residue list file!

Debugging Crashes Part II:

Things that will generally make your life miserable…

Dumb mistakes:
Did you set the Read and Multi flags correctly? If not, → CRASH!!!
Did you request cPCA but not specify a Cartesian residue list? If so, → CRASH!!!
Did you set the number of modes correctly? If not, → CRASH!!!
Are the 0 and 1 flags set correctly? If not → CRASH!!!
Are you requesting to read PDBs when you should be specifying a coordinate matrix? If so → CRASH!!!

More subtle mistakes:
Did you request more modes than actually exist? For example, if your Cartesian subset contains 12 residues, but you asked for 50 modes, then you are going to crash JED because there are only 36 Cartesian modes in total.

Also, if your Distance subset contains 3 residues and you request 5 modes, then you are going to crash JED because there are only 3 distance modes in total.

If your trajectory has not equilibrated, then you must address the problem or outliers. If you do not, then the Q and R matrices will be highly ill-conditioned and may cause the eigenvalue decomposition to fail. You can check the variables in statistics packages that compute the Kaiser-Myer-Olkin (KMO) statistic as well as the Measure of Sampling Adequacy (MSA) for each coordinate variable to critically assess your data. If it is not well suited for PCA, you can condition the variables by setting the z-cutoff in JED between 2.0 and 3.0 when running your jobs. This type of conditioning is by far not very sophisticated, but it has the effect of lowering the condition numbers of Q and R as well as un-dilating the high and low regions of the eigenspectrum. In particular, it does not alter the ordinality of the eigenspectrum but does correct the distortion that arises from under sampling when trying to estimate the population covariance matrix from the sample covariance matrix.

Performing Cartesian and Distance PCA with Mode Visualization

The working directory must contain: The coordinates matrix, the PDB reference file, and both residue lists. It may also contain JED_Driver.txt

JED is capable of doing both cPCA and dPCA, using both Q and R, and generating cPCA mode visualizations simultaneously. All outputs are delivered as discussed for the individual components.

The outputs to this type of job include the outputs for both the cPCA and dPCA analyses, as well as all the structures for the top modes chosen for visualization. JED will permute the reference structure for a given subset along the top eigenvectors selected for visualization and output structures (PDBs) that capture one cycle of this motion. The amplitude of the motion is determined by the value of the **$scale_factor**, whose default value is 1.0, and can be adjusted as necessary. Setting the value too high will cause Visualization software like Pymol to break the ribbon diagrams of the structures. Ultimately, this is controlled by the magnitude of the eigenvector components for any given residue. Setting the scale factor between 1 and 3 is usually safe, but for proteins with highly mobile regions like loops, you may need to choose a smaller scale factor. This is done for both the top Q and R modes. Additionally, Pymol scripts are generated to animate those structures into a movie for better analysis of the physical meanings of the top modes.

These files will be located in the /mode-viz subdirectory of the root of the JED results tree:
`/working/directory/JED_Results_Description/mode_viz/`

The Q results will be in the subdirectory /COV and the R results will be in the subdirectory /CORR.

One huge advantage of JED is that it is highly configurable and can perform many types of Essential Dynamics analysis concurrently. Combined with the cluster resources or just using the batch feature allows a user to process a great deal of data efficiently.

Below is a sample input file for Single-Chain PDB files:

```
-------------------------------------------------------------------------------
0.00
3.00
0     0
20    3     2     1.0
/working/directory/
Description     reference_PDB_file.pdb
residues_cartesian.txt
residues_dist.txt
original_PDB_Coordinates.txt 0
-------------------------------------------------------------------------------
```

Notes:

Line 1 specifies the **percent** (double) of frames to remove from the data: 0 = none

Line 2 specifies the **z-cutoff** (double) for adjusting outliers: 0 → values beyond |3.0|

Line 3 Token 1 specifies whether to **read PDB files** (0 or 1) → 0 = no

Line 3 Token 2 specifies if the PDB files are **Multi** Chain (0 or 1) → 0 = no

Line 4 Token 1 specifies the **number of Cartesian** (integer) modes to process → 0 = none

Line 4 Token 2 specifies the **number of Distance** (integer) modes to process → 3 = top three modes

Line 4 Token 3 specifies the **number of Cartesian modes to Visualize**(integer) → 0 = none

Line 5 specifies the **working directory** (String)

Line 6 Token 1 specifies the job **description** (String)

Line 6 Token 2 specifies the **reference PDB** (String)

Line 7 specifies the **Cartesian residue list** (String)

Line 7 specifies the **Distance residue list** (String)

Line 8 Token 1 specifies the **coordinate matrix** (String)

Line 8 Token 2 specifies the **reference column** (integer)

Warning:

Both the read PDB file flag and the Multi flag MUST be set to ZERO.

The number of cPCA modes MUST be > zero.

The number of dPCA modes MUST be > zero.

There must be a Cartesian residue list specified.

There must be a Distance residue list specified.

They must be in the CORRECT ORDER: Cartesian first, then Distance

For Cartesian subsets containing ***N*** residues, you must NOT request more than **3*N*** modes.

For Distance subsets containing ***N*** residues, you must NOT request more than ***N(N-1)/2*** modes.

Below is a sample input file for Multi-Chain PDB files:

Differences from Single-Chain analysis shown highlighted in amber

```
--------------------------------------------------------------------------------
0.00
3.00
0     1     2     A     B     795     151     0     0
20    3     2     1.0
/working/directory/
Description     reference_PDB_file.pdb
residues_cartesian.txt
residues_dist.txt
original_PDB_Coordinates.txt        0
--------------------------------------------------------------------------------
```

Notes:

Line 1 specifies the **percent** (double) of frames to remove from the data: 0 → None

Line 2 specifies the **z-cutoff** (double) for adjusting outliers: 0 → values beyond |3.0|

Line 3 Token 1 specifies whether to **read PDB files** (0 or 1) → 0 = no

Line 3 Token 2 specifies if the PDB files are **Multi** Chain (0 or 1) → 1 = yes

Line 3 Token 3 specifies the number of chains → (2);

Line 3 Token 4 specifies the first chain ID → (A);

Line 3 Token 5 specifies the second chain ID → (B);

Line 3 Token 6 specifies the number of residues in chain A→ (795);

Line 3 Token 7 specifies the number of residues in chain B→ (151);

Line 3 Token 8 specifies the offset of Chain A → (0);

Line 3 Token 9 specifies the offset of Chain B → (0);

Line 4 Token 1 specifies the **number of Cartesian** (integer) modes to process → 20 = top twenty modes

Line 4 Token 2 specifies the **number of Distance** (integer) modes to process → 3 = top three modes

Line 4 Token 3 specifies the **number of Cartesian modes to Visualize**(integer) → 2 = top two modes

Line 5 specifies the **working directory** (String)

Line 6 Token 1 specifies the job **description** (String)

Line 6 Token 2 specifies the **reference PDB** (String)

Line 7 specifies the **Cartesian residue list** (String)

Line 7 specifies the **Distance residue list** (String)

Line 8 Token 1 specifies the **coordinate matrix** (String)

Line 8 Token 2 specifies the **reference column** (integer)

Warning:

The read PDB file flag MUST be set to ZERO.

The multi flag must be set to ONE; This tells JED to perform the analysis on MULTI Chain PDBs.

The number of cPCA modes MUST be > zero.

The number of dPCA modes MUST be > zero.

There must be a Cartesian residue list specified.

There must be a Distance residue list specified.

They must be in the CORRECT ORDER: Cartesian first, then Distance

For Cartesian subsets containing ***N*** residues, you must NOT request more than **3*N*** modes.

For Distance subsets containing ***N*** residues, you must NOT request more than ***N(N-1)/2*** modes.

Remember that the format of the residue list file is TWO columns for Multi Chain Analysis: **Chain ID, residue number** while for Single Chain Analysis, the file has ONE column: **residue number**.

If number of cPCA modes > 0, then there must be a Cartesian residue list file!

If number of dPCA modes > 0, then there must be a Distance residue list file!

Additional Types of Analysis

Pooling Data:

It is often useful to pool trajectory statistics. This can be done in JED by combining coordinate files and then performing the usual analysis. To combine the coordinate files, there is a utility program called **JED_Pool_Data.java** that will combine multiple matrices into one. Of courses, the number of rows in the coordinate files must match. The matrices to combine are specified by an input file called **pool.txt** that the user must construct correctly.

Below is a sample pool.txt file:

```
-------------------------------------------------------------------------------------------
1
3
/output/directory/
/path/to/first/matrix/matrix_1.txt
/path/to/first/matrix/matrix_2.txt
/path/to/first/matrix/matrix_3.txt
-------------------------------------------------------------------------------------------
```

Notes:

Line #1 specifies the number of jobs (integer) →1

Then for each job you must specify the following:

Line #2 specifies the number of matrices to combine (integer) →3

Line #3 specifies the output directory (string)

Line #4 specifies the path to the first matrix (String)

Line #5 specifies the path to the second matrix (String)

Line #6 specifies the path to the third matrix (String)

Subspace Analysis:

Once JED Driver has been run on multiple trajectories as well as pooled trajectories, an analysis can be done to compare how similar the essential subspaces derived from those trajectories are to each other. JED contains a program called **Subspace_Analysis.java** along with 3 driver programs that perform those functions. The core program takes as input two matrices of eigenvectors derived from PCA (or NMA, ANM, etc.). The matrices must have the same number of rows and columns, meaning the vectors being compared come from the same vector space and that the subspaces have the same dimensions. For example, in an analysis of lysozyme you might choose to process 20 cPCA modes while examining 10 different experimental conditions plus pooled data. As long as all the subsets in the analysis are the same then all the 20 dimensional subspaces can be compared.

Like most of the JED programs, the subspace analysis program driver reads an input file called **SSA.txt** to obtain runtime information. This file must be constructed properly by the user to perform the analysis. The three driver programs are **SSA_Driver.java**, **FSSA_Driver.java**, and **FSSA_Iterated_Driver.java** and are different in how much analysis is requested. The SSA_Driver gives full outputs for non-iterated subspace comparison including both log files and individual flat files. The FSSA_Driver is a light-weight version with only RMSIP and PA output in the log files. The Iterated version performs a recursive variation of the above where all equidimensional subspaces are compared up to the size that was provided, for example, from 1 to 20 by step-size 1 for a 20 column input file.

ALL three drivers use the same input file, only the outputs are different.

Below is a sample subspace_analysis.txt file:

The format for the file shown below is:

LINE 1: Number_of_Jobs (integer)
LINE 2: Output_Directory (string ending in "/" or "\\")
LINE 3: Batch_Decription (string)
FOR EACH JOB:
Description (string)
Directory1 (string ending in "/" or "\\")   Name1 (string)
Directory2 (string ending in "/" or "\\")   Name2 (string)

```
---------------------------------------------------------------------------------------------
4
/output/directory/
Single_Combo_SSA
All-vs-A
/Users/physicslabs/                    all_combo_SS_75_top_20_eigenvectors.txt
/Users/physicslabs/                    1a6n_combo_SS_75_top_20_eigenvectors.txt
All-vs-B
/Users/physicslabs/                    all_combo_SS_75_top_20_eigenvectors.txt
/Users/physicslabs/                    1wit_combo_SS_75_top_20_eigenvectors.txt
All-vs-A+B
/Users/physicslabs/                    all_combo_SS_75_top_20_eigenvectors.txt
/Users/physicslabs/                    1ubq_combo_SS_75_top_20_eigenvectors.txt
All-vs-A_B
/Users/physicslabs/                    all_combo_SS_75_top_20_eigenvectors.txt
/Users/physicslabs/                    1ypi_combo_SS_75_top_20_eigenvectors.txt
---------------------------------------------------------------------------------------------
```

It is possible to specify hundreds of comparisons when using the driver input file, but please construct the input file carefully... Just one typo in a list of one thousand comparisons → CRASH.