<u>Introduction</u>

Java Essential Dynamics is a set of java programs for analyzing protein trajectories. The trajectories may be derived from MD, FIRST/FRODA, or 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 aligns all the structures in the trajectory to a specified reference structure using a quaternion alignment algorithm. The **conformation rmsd** is determined for each structure using a specified reference structure and the **residue rmsd** is determined from the entire trajectory. An **edited PDB file** is generated for the specified subset 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 using the JED_Driver.java program. The first and most common method is a Cartesian coordinate based analysis (cPCA)using the selected set of the alpha carbons. The second method is an internal coordinate based analysis using the distances (dPCA) between each residue 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 Cartesian and distance modes to determine, 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 Pymol script is generated to animate the frames. Note that dPCA modes cannot be mapped to a structure in an intuitive way.

The two methods 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. The program computes the **PCA modes** (RMSD and MSD) from the Cartesian eigenvectors so that they may be mapped to the residue set. Additionally, 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 (these are the principle components (PCs)). These **DV projections** (DVP), also known as the principle components (PCs) allow the user to investigate how the trajectory projects into the space defined by each eigenvector. The DVPs are given using un-normalized and normalized inner products as well as weighted by the corresponding eigenvalue.

JED performs a simple subspace analysis on the two sets of eigenvectors generated from the Q and R PCA. This allows one to determine how different the PCA results are due only to the choice of PCA model. Additional analysis can be done using the Subspace_Analysis.java program. To perform these tests, one first needs to generate sets of eigenvectors from each trajectory as well as from pooled trajectories. Subspace analysis is done by comparing the sets of eigenvectors and determining the RMSIP, Principal Angles (PA), cumulative overlap

(CO), cosine products (CP), vectorial angular sum (VAS), and the maximum angle between subspaces of the given size.

<u>**Running the Program**</u>

JED is a java based set of programs and as such can run on any platform with a suitable JDK.

The Cobra computer cluster has a 64 bit JDK installed. Be sure to request adequate memory resources as matrix diagonalization scales as O^3 (12-36GB) or use BigMem if necessary.

Each job should be assigned in its own directory, which should contain either the PDB files to read or the coordinate file to process, and the reference PDB file.

A top level directory should contain the residue file lists for specifying the Cartesian and distance subsets.

NOTE: A preliminary run with no PCA should be run using all residues in the protein to generate a coordinate file for all the alpha carbons. This makes multiple subset analyses much more efficient as the program only needs to read in the PDB files once.

<u>**The files that specify the residue lists for the Cartesian and Distance subsets are formatted as follows:**</u>

Single chain PDBs use a single column: Residue Number

Multi-Chain PDBs use a double column: Chain ID, Residue Number

The main JED program is called **JED_Driver.java**

This program obtains all run-time information from an input file that is read when the program is run:

The JED_Driver input file is called **JED.txt**

It is critical that this file be constructed correctly and be located in the directory where the JVM is called.

The file is tab (space) delimited .

The file will have different number of lines/tokens depending on the options selected:

<u>The key differentiating factors are:</u>

Will the program will read in all the PDB files in the working directory or read in a coordinate file?

Are the PDB files single chain or multi chain?

Will the analysis do cPCA, dPCA, BOTH, or NONE?

Will Cartesian mode visualization be done?

(In the following, code and text file content is shown in `10 point Consolas`**)**

Below is a sample JED.txt file for Single Chain PDB files: Both cPCA & dPCA

The batch consists of 2 jobs , will read PDB files, and will perform both Cartesian and Distance subset analyses.

The top 20 cPCA modes, and the top 3 dPCA modes will be generated;

The top 5 cPCA modes will processed for mode visualization.

```
----------------------------------------------------------------------------------------
2
1     0
20     3     5     1.0
/dir/to/Cartesian/residue/list/  residue_list_cart_filename.txt   0
/dir/to/distances/residue/list/  residue_list_dist_filename.txt   0
**************************************************************************
/working/directory/for/job/1/    Description1  reference_PDB_file1.pdb
/working/directory/for/job/2/    Description2  reference_PDB_file2.pdb

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

Notes:
Line #1 specifies the number of jobs (integer)
Line #2 Token #1 specifies whether to read PDB files (0 or 1),
Line #2 Token #2 specifies if the PDB files are Multi Chain (0 or 1);
Line #3 Token #1 specifies the number of Cartesian modes to process (integer)
Line #3 Token #2 specifies the number of Distance modes to process (integer)
Line #3 Token #3 specifies the number of Cartesian modes to Visualize (integer)
Line #3 Token #4 specifies the scale factor for the visualization (double)
Line #4 Token #1 specifies the path to the Cartesian Subset Residue List (String)
Line #4 Token #2 specifies the name of the Cartesian Subset Residue List (String)
Line #4 Token #3 specifies the chain offset (integer)
Line #5 Token #1 specifies the path to the Distance Subset Residue List (String)
Line #5 Token #2 specifies the name of the Distance Subset Residue List (String)
Line #5 Token #3 specifies the chain offset (integer)
Line #6 is a separator line: header/batch info above, and job listings below.
Line #7 Token #1 specifies the working directory(WD) for job 1 (String)
Line #7 Token #2 specifies the description for job 1 (String)
Line #7 Token #3 specifies the reference PDB file for job 1 (String)
Line #8 Token #1 specifies the working directory(WD) for job 1 (String)
Line #8 Token #2 specifies the description for job 1 (String)
Line #8 Token #3 specifies the reference PDB file for job 1 (String)

Gotchas! (things that will generally make your life miserable…)

Does the file start on the first line? If not, → CRASH!!!
Is dPCA specified but no residue list is specified for the distance subset? If so, → CRASH!!!
The WD contains PDB files of different sizes → CRASH!!!
The WD does not contain the reference PDB file → CRASH!!!
Is the integer an Integer or a float? → CRASH!!!
Does the directory string end in "/" or "\\"?: If not → CRASH!!!
Number of cPCA residues = 3, If number of cPCA modes > 3*3 = 9 → CRASH!!!

Number of dPCA residues = 3, If number of dPCA modes > 3*2/2 = 3 → CRASH!!!
Number of cPCA modes = 1, modes to visualize = 3 → CRASH!!!
The WD contains PDB files from mode visualizations: Move to a sub-directory!
The WD contains PDB files with 2 chains but no chain IDs → Non-sense results depending on subset
If the PDB file numbering starts at 5, then the chain offset is not 0, its 4 → Garbage out!

IN SUMMARY:
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 designed to do exactly this)

Below is a sample JED.txt file for Single Chain PDB files: THE PRELIMINARY RUN

The batch consists of 2 jobs , will read PDB files, and will perform NO PCA.

The purpose of this is to generate the matrix of coordinates for performing subset analyses efficiently.

For this to work, the residue list should contain ALL the residues in the protein (or chain of interest).

```
------------------------------------------------------------------------------------------
2
1      0
0      0      0      1.0
/dir/to/Cartesian/residue/list/  all_residues.txt    0
**************************************************************************
/working/directory/for/job/1/    Description1  reference_PDB_file1.pdb
/working/directory/for/job/2/    Description2  reference_PDB_file2.pdb

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

Notes:
Line #1 specifies the number of jobs (integer) →2
Line #2 Token #1 specifies whether to read PDB files (0 or 1) →1=yes
Line #2 Token #2 specifies if the PDB files are Multi Chain (0 or 1) →0=no
Line #3 Token #1 specifies the number of Cartesian modes to process (integer) →0=none
Line #3 Token #2 specifies the number of Distance modes to process (integer) →0=none
Line #3 Token #3 specifies the number of Cartesian modes to Visualize (integer) →0=none
Line #3 Token #4 specifies the scale factor for the visualization (double) → N/A
Line #4 Token #1 specifies the path to the Cartesian Subset Residue List (String)
Line #4 Token #2 specifies the name of the Cartesian Subset Residue List (String) → Specify ALL residues
Line #4 Token #3 specifies the chain offset (integer)
Line #5 is a separator line: header/batch info above, and job listings below.
Line #6 Token #1 specifies the working directory(WD) for job 1 (String)
Line #6 Token #2 specifies the description for job 1 (String)
Line #6 Token #3 specifies the reference PDB file for job 1 (String)
Line #7 Token #1 specifies the working directory(WD) for job 2 (String)
Line #7 Token #2 specifies the description for job 2 (String)
Line #7 Token #3 specifies the reference PDB file for job 2 (String)

Output Files:
JED Log and PDB Read Log, Original and Transformed PDB coordinates, conformation and residue rmsd.

Below is a sample JED.txt file for Single Chain PDB files: cPCA only

The batch consists of 2 jobs , will read coordinates from files, and will perform only cPCA.

The top 20 modes will be processed, but no mode visualization will be done.

```
-------------------------------------------------------------------------------------------
2
0     0
20     0     0     1.0
/dir/to/Cartesian/residue/list/  subset_residues.txt    0
*************************************************************************
/working/directory/for/job/1/    Description1  reference_PDB_file1.pdb
PDB_coordinates1.txt              0
/working/directory/for/job/2/    Description2  reference_PDB_file2.pdb
PDB_coordinates2.txt              0

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

Notes:
Line #1 specifies the number of jobs (integer) →1
Line #2 Token #1 specifies whether to read PDB files (0 or 1) →0=no
Line #2 Token #2 specifies if the PDB files are Multi Chain (0 or 1) →0=no
Line #3 Token #1 specifies the number of Cartesian modes to process (integer) →20
Line #3 Token #2 specifies the number of Distance modes to process (integer) →0=none
Line #3 Token #3 specifies the number of Cartesian modes to Visualize (integer) →0=none
Line #3 Token #4 specifies the scale factor for the visualization (double) → N/A
Line #4 Token #1 specifies the path to the Cartesian Subset Residue List (String)
Line #4 Token #2 specifies the name of the Cartesian Subset Residue List (String) → Specify subset
Line #4 Token #3 specifies the chain offset (integer)
Line #5 is a separator line: header/batch info above, and job listings below.
Line #6 Token #1 specifies the working directory(WD) for job 1 (String)
Line #6 Token #2 specifies the description for job 1 (String)
Line #6 Token #3 specifies the reference PDB file for job 1 (String)
Line #7 Token #1 specifies the name of the coordinates file for job 1 (String)
Line #7 Token #2 specifies the reference column in the coordinates matrix for job 1 (integer)
Line #8 Token #1 specifies the working directory(WD) for job 2 (String)
Line #8 Token #2 specifies the description for job 2 (String)
Line #8 Token #3 specifies the reference PDB file for job 2 (String)
Line #9 Token #1 specifies the name of the coordinates file for job 2 (String)
Line #9 Token #2 specifies the reference column in the coordinates matrix for job 2 (integer)

When reading from coordinate files, TWO lines are necessary to specify a job.

Same gotchas apply here…make sure the files and paths exist!

Next, we consider how to handle multi chain PDB files.

<u>Below is a sample JED.txt file for Multi Chain PDB files: THE PRELIMINARY RUN</u>

The batch consists of 2 jobs , will read PDB files, and will perform both Cartesian and Distance subset analyses.

The top 20 cPCA modes, and the top 3 dPCA modes will be generated;

The top 5 cPCA modes will processed for mode visualization.

```
------------------------------------------------------------------------------------------
2
1     1     2     A     B     795     151     0     0
0     0     0     1.0
/dir/to/Cartesian/residue/list/  all_residues.txt
*************************************************************************
/working/directory/for/job/1/    Description1  reference_PDB_file1.pdb
/working/directory/for/job/2/    Description2  reference_PDB_file2.pdb

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

<u>Notes</u>:
Line #1 specifies 2 jobs (2)
Line #2 Token #1 specifies whether to read PDB files (1),
Line #2 Token #2 specifies if the PDB files are Multi Chain (1);
Line #2 Token #3 specifies the number of chains (2);
Line #2 Token #4 specifies the first chain ID (A);
Line #2 Token #5 specifies the second chain ID (B);
Line #2 Token #6 specifies the number of residues in chain A (795);
Line #2 Token #7 specifies the number of residues in chain A (151);
Line #2 Token #8 specifies the offset of Chain A (0);
Line #2 Token #9 specifies the offset of Chain B (0);
Line #3 Token #1 specifies no Cartesian modes to process (0)
Line #3 Token #2 specifies no Distance modes to process (0)
Line #3 Token #3 specifies no Cartesian modes to Visualize (0)
Line #3 Token #4 specifies the scale factor for the visualization (N/A)
Line #4 Token #1 specifies the path to the Cartesian Subset Residue List (String)
Line #4 Token #2 specifies the name of the Cartesian Subset Residue List (All residues for this analysis)
Line #5 is a separator line: header/batch info above, and job listings below.
Line #6 Token #1 specifies the working directory(WD) for job 1 (String)
Line #6 Token #2 specifies the description for job 1 (String)
Line #6 Token #3 specifies the reference PDB file for job 1 (String)
Line #7 Token #1 specifies the working directory(WD) for job 2 (String)
Line #7 Token #2 specifies the description for job 2 (String)
Line #7 Token #3 specifies the reference PDB file for job 2 (String)

<u>Gotchas!</u>

Make sure that the chain information is correct or the output will be non-sense!

Make sure that file paths and names are correct!

<u>Below is a sample JED.txt file for Multi Chain PDB files: cPCA only</u>

The batch consists of 2 jobs , will read coordinates from files, and will perform only cPCA.

The top 20 modes will be processed, but no mode visualization will be done.

```
------------------------------------------------------------------------------------------
2
0     1     2     A     B     795     151     0     0
20     0     0     1.0
/dir/to/Cartesian/residue/list/  subset_residues.txt
*************************************************************************
/working/directory/for/job/1/    Description1  reference_PDB_file1.pdb
PDB_coordinates1.txt            0
/working/directory/for/job/2/    Description2  reference_PDB_file2.pdb
PDB_coordinates2.txt            0

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

<u>Notes</u>:
Line #1 specifies the number of jobs (integer) →2
Line #2 Token #1 specifies whether to read PDB files (0 or 1) →0=no
Line #2 Token #2 specifies if the PDB files are Multi Chain (0 or 1) →0=no
Line #3 Token #1 specifies the number of Cartesian modes to process (integer) →20
Line #3 Token #2 specifies the number of Distance modes to process (integer) →0=none
Line #3 Token #3 specifies the number of Cartesian modes to Visualize (integer) →0=none
Line #3 Token #4 specifies the scale factor for the visualization (double) → N/A
Line #4 Token #1 specifies the path to the Cartesian Subset Residue List (String)
Line #4 Token #2 specifies the name of the Cartesian Subset Residue List (String) → Specify subset
Line #4 Token #3 specifies the chain offset (integer)
Line #5 is a separator line: header/batch info above, and job listings below.
Line #6 Token #1 specifies the working directory(WD) for job 1 (String)
Line #6 Token #2 specifies the description for job 1 (String)
Line #6 Token #3 specifies the reference PDB file for job 1 (String)
Line #7 Token #1 specifies the name of the coordinates file for job 1 (String)
Line #7 Token #2 specifies the reference column in the coordinates matrix for job 1 (integer)
Line #8 Token #1 specifies the working directory(WD) for job 2 (String)
Line #8 Token #2 specifies the description for job 2 (String)
Line #8 Token #3 specifies the reference PDB file for job 2 (String)
Line #9 Token #1 specifies the name of the coordinates file for job 2 (String)
Line #9 Token #2 specifies the reference column in the coordinates matrix for job 2 (integer)

Recall that when reading from coordinate files, TWO lines are necessary to specify a job.

Same gotchas apply here…make sure the files and paths exist!

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 a single column: residue number.

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

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 in the toolkit package called **Create_Augmented_Matrix.java** that will combine multiple matrices into one. Of courses, the dimensions of the coordinate files must match. The matrices to combine are specified by an input file called **augmented_matrix.txt** that the user must construct correctly.

Below is a sample augmented_matrix.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 Token #1 specifies the path to the first matrix (String)
Line #4 Token #2 specifies the name of the first matrix(String)
Line #5 Token #1 specifies the path to the second matrix (String)
Line #5 Token #2 specifies the name of the second matrix(String)
Line #6 Token #1 specifies the path to the third matrix (String)
Line #6 Token #2 specifies the name of 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 has a program called **Subspace_Analysis.java** along with 3 driver programs that performs those functions. The core program takes as input two matrices of eigenvectors derived from prior PCA. 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 **subspace_analysis.txt** to obtain runtime information. This file must be constructed properly by the user to perform the analysis. The 3 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 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 3 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 beware of the usual gotchas. One typo in a list of one thousand comparisons → CRASH.