ADEPT User's Manual

Jeffrey Perez

August 24, 2012

Contents

1 Introduction

1.1 About

The **A**utomatic **D**ata **E**xtraction and **P**lotting **T**ool (ADEPT) is a Java application which is used to help in the analysis of data output from QUEST simulations [1]. Output files from QUEST - containing normal user-readable data - are parsed into variables in an XML-like structure which allow ADEPT to access the data in a more organized way.When the files are passed into ADEPT, it opens a Graphical User Interface (GUI) allowing the user to import, manipulate, and save data from numerous QUEST output files into a format which can be easily plotted using the user's choice of plotting software.

1.2 Motivation

ADEPT was created based on a need which arises from comparing and plotting data from multiple QUEST simulations. Previously, the required data would have to be manually extracted by the user either with a copy-paste approach or using a parsing script, each of which had difficulties associated with them.

- The copy-paste approach is obviously very labor-intensive when the user wishes to compare data from a large number of files.
- Although parsing scripts do allow for extraction of data, this approach still requires significant work from the user if any manipulation of this data is needed.

Another need that arose during development of the application was the ability to start with a large repository of data files and only compare files with certain values for certain data elements. For example, one may only want data from simulations where the interaction strenght U is equal to 4. In this case, the parsing approach would be of no help, unless the data files were themselves named using some strict labeling convention.

2 Building ADEPT

2.1 Basic Install

ADEPT's only requirement to run is Java version 1.7.0, available at
`http://www.oracle.com/technetwork/java/javase/downloads/java-se-jre-7-download-432155.html` You can check your version of Java with the command `java -version`. The output should look something like:

```
$ java -version
java version "1.7.0_06"
Java(TM) SE Runtime Environment (build 1.7.0_06-b24)
Java HotSpot(TM) 64-Bit Server VM (build 23.2-b09, mixed mode)
```

2.2 Running ADEPT

ADEPT is run from the command line using the the -jar option and offers a number of acceptable inputs as outlined below.

2.2.1 Without command line arguments

The most basic command to run ADEPT is:

```
java -jar ADEPT.jar
```

This will open a file chooser dialog where the user must choose at least one valid input file to initialize the variable menus, which are explained more in 3.4.1. This method is best when the user has multiple files they wish to open which may not have a common naming convention. Although files selected during initialization in this way are opened from the same directory, the user may add more files from any other directory once the application is open, as seen in 3.1.1.1.

2.2.2 With file names

Instead of using the file chooser, the user may also input files directly into the command line. This would be a better choice if the user wishes to input files according to some naming convention - or simply all files. The following commands are all valid examples of this method of calling ADEPT.

```
java -jar ADEPT.jar *.txt
java -jar ADEPT.jar file1.txt file2.txt
java -jar ADEPT.jar ./testfiles/file1.txt file*.txt
```

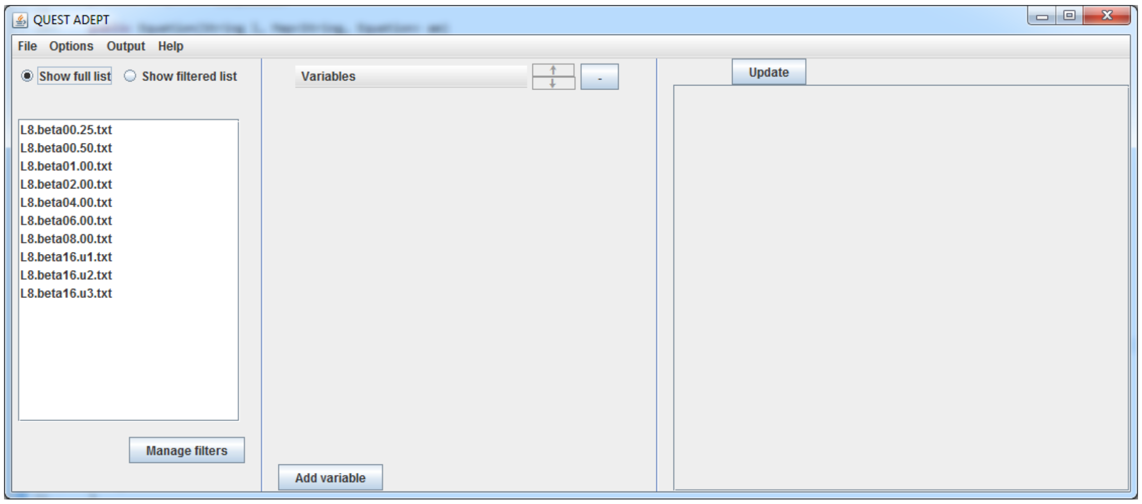
2.2.3 With variable lists and file names

Finally, the user may define a special *variable list* file which defines lists of variables the user wishes to choose from during execution. A detailed explanation and how to define this file is described in 3.4.2, but for now we will assume this file is already created. The command includes the path to the variable list file as the first argument.

```
java -jar ADEPT.jar ./variableLists/variableList.xml *.txt
```

3 GUI Overview

This section will go over every element of the interface in detail. To start, the main window upon execution is shown below and was called with the included `variablelists.xml` and a selection of sample data files.



L8.beta00.25.txt
L8.beta00.50.txt
L8.beta01.00.txt
L8.beta02.00.txt
L8.beta04.00.txt
L8.beta06.00.txt
L8.beta08.00.txt
L8.beta16.u1.txt
L8.beta16.u2.txt
L8.beta16.u3.txt

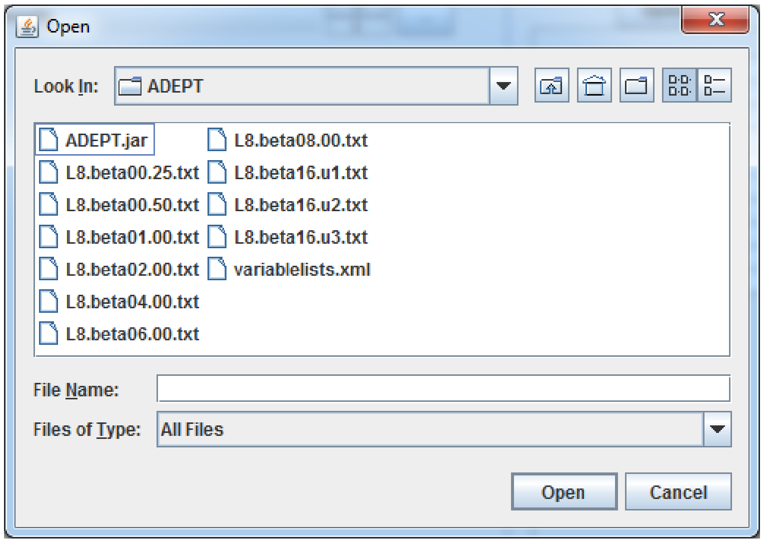
3.1 Menu Bar

The menu bar at the top of the window contains elements used for file managment, data options, preset output sets, and help.

3.1.1 File

3.1.1.1 Add file

Files may be added to the full file list during execution of ADEPT. These files will also be added to the filtered file list if those files pass through each user-defined filter. For information on filtering, see 3.3. The window for adding files is displayed below. This file chooser is also what is used when no command line arguments are provided, to choose a file for initialization.



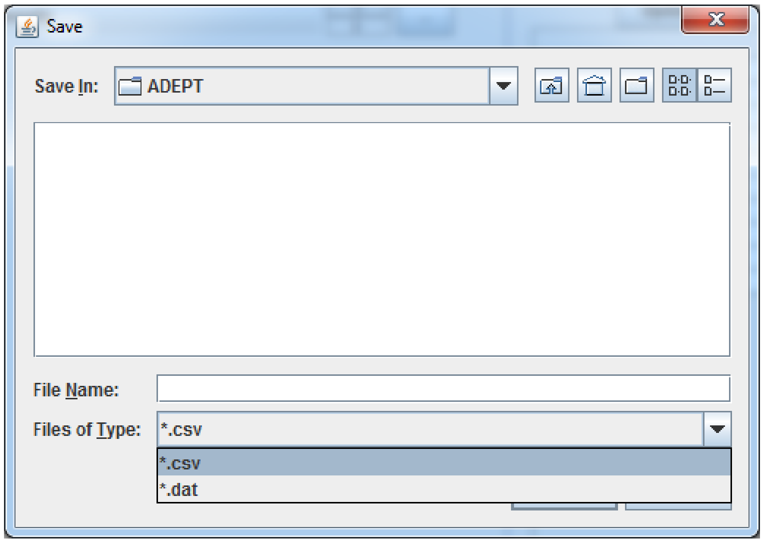
ADEPT.jar
L8.beta00.25.txt
L8.beta00.50.txt
L8.beta01.00.txt
L8.beta02.00.txt
L8.beta04.00.txt
L8.beta06.00.txt
L8.beta08.00.txt
L8.beta16.u1.txt
L8.beta16.u2.txt
L8.beta16.u3.txt
variablelists.xml

3.1.1.2 Save as...

Once results are displayed in the result panel, the user may save the entries to a file, which may be used for easy plotting. Currently, two file formats are supported to save as: .csv and .dat.

- Comma Separated Value (CSV) files will contain the data exactly as shown in the result panel, with rows separated by line breaks and columns separated by commas.
- DAT files also separate rows by line breaks, but separate columns by tabs. This format is usable in plotting utilities like gnuplot.

Supported file formats can be seen in the drop down menu below.



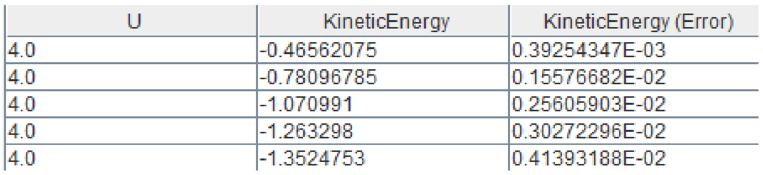
3.1.1.3 Exit

This option exits ADEPT and its child windows.

3.1.2 Options

3.1.2.1 Include value errors in results

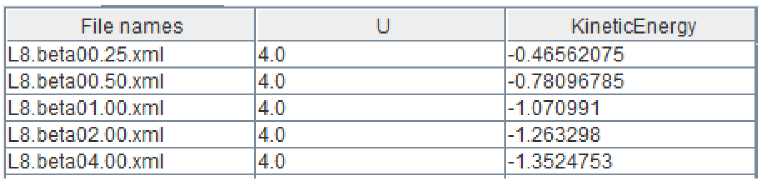
Some data elements in QUEST output files have certain error values attached to them. By default these error values are not shown, but by enabling this option the error values will appear in a new column next to the data element they are associated with. This only applies to data elements with an error associated with them; other elements are displayed normally.



U	KineticEnergy	KineticEnergy (Error)
4.0	-0.46562075	0.39254347E-03
4.0	-0.78096785	0.15576682E-02
4.0	-1.070991	0.25605903E-02
4.0	-1.263298	0.30272296E-02
4.0	-1.3524753	0.41393188E-02

3.1.2.2 Include file names in results

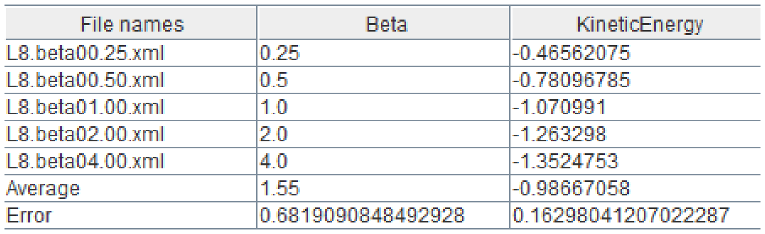
Ordinarily, rows in the result panel are written in same order as the file names in the file list panel. When looking at a large amount of results however, it can be confusing which row of values belongs to which file. This option adds a column on the left of the result panel containing the file names for each row.



File names	U	KineticEnergy
L8.beta00.25.xml	4.0	-0.46562075
L8.beta00.50.xml	4.0	-0.78096785
L8.beta01.00.xml	4.0	-1.070991
L8.beta02.00.xml	4.0	-1.263298
L8.beta04.00.xml	4.0	-1.3524753

3.1.2.3 Include average values and errors in last row

In the case that the user wishes to know the average value or error of a data element across all files, this option adds two rows to the bottom of the result panel - the average and error respectively. If it is unclear which row is the average, which is the error, and which is a normal data value, then the user may enable the **Include file names in results** option which will label these rows accordingly. Since the average and error rows are not derived from one single file, their file names are simply "Average" and "Error".



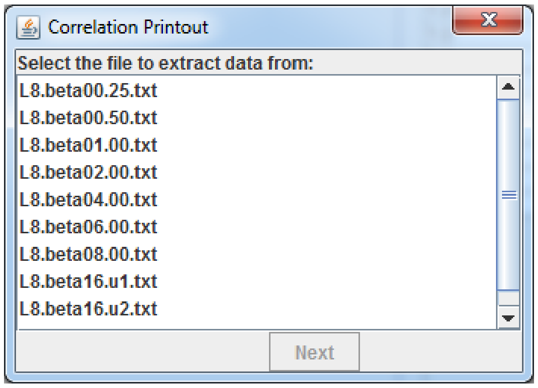
File names	Beta	KineticEnergy
L8.beta00.25.xml	0.25	-0.46562075
L8.beta00.50.xml	0.5	-0.78096785
L8.beta01.00.xml	1.0	-1.070991
L8.beta02.00.xml	2.0	-1.263298
L8.beta04.00.xml	4.0	-1.3524753
Average	1.55	-0.98667058
Error	0.6819090848492928	0.16298041207022287

3.1.2.4 Show ____ variables

Variable lists - either default or user-defined - can be set visible or invisible to the user when searching for variables in the menus. For example, the Extended variable list is turned invisible by default when another variable list is defined. Creating user-defined variable lists and setting its default visibility and ability to toggle the visibility are covered in 3.4.2.

3.1.3 Output

The Output menu contains specially constructed output formats based on a normal QUEST output. Currently, these outputs extract data from one file and one correlation function within that file. The file choose window is displayed below.


Correlation Printout

Select the file to extract data from:

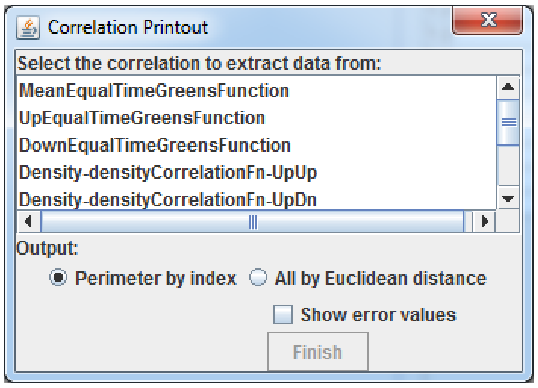
L8.beta00.25.txt
L8.beta00.50.txt
L8.beta01.00.txt
L8.beta02.00.txt
L8.beta04.00.txt
L8.beta06.00.txt
L8.beta08.00.txt
L8.beta16.u1.txt
L8.beta16.u2.txt

3.1.3.1 Correlation Printout

The user can obtain the data values for numerous points in a correlation with one option. With this option, the user may output the perimeter of the correlation graph with the indexing scheme described below, or output all points by Euclidean distance from the origin.

- Perimeter by index: These values start along the x-axis as defined by $Dx = 0$, continue up the far edge where $Dx = max(Dx)$, and finally along the hypotenuse where $Dx = Dy$. If the perimeter is viewed as a right triangle with the $(max(Dx), max(Dy))$ is on the top right, the origin at the bottom left and $(max(Dx), 0)$ at the bottom right, the indices of this perimeter could be found by tracing the triangle in a counter-clockwise direction starting at the origin.
- All by Euclidean distance: Extracts values from all points from the correlation, outputting the Euclidean distance from the origin and the value of the correlation at that distance.

Additionally, there is an option to show the error values for each point extracted.


Correlation Printout

Select the correlation to extract data from:

MeanEqualTimeGreensFunction
UpEqualTimeGreensFunction
DownEqualTimeGreensFunction
Density-densityCorrelationFn-UpUp
Density-densityCorrelationFn-UpDn

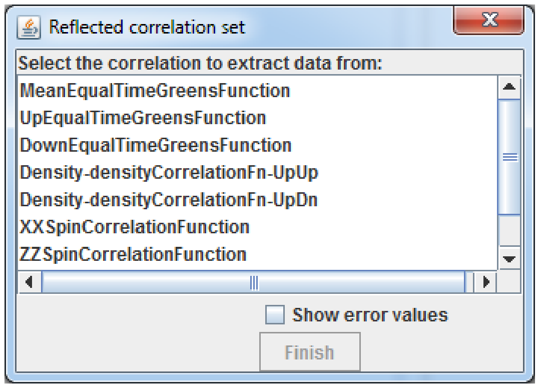
Output:

[✓ Perimeter by index] All by Euclidean distance

Show error values

3.1.3.2 Whole correlation set

When plotting a correlation on a contour plot, it is helpful to have the entire graph, reflected across the $X = Y$, 0, and $Y = 0$ axes. This output takes the data from a correlation, performs this reflection, and places all points into the result panel. In this way, the user may save the data to a file and easily plot the entire graph without having to do the reflection themselves. This output simply has the user pick a correlation. No special indexing is performed; indexing is pulled from the data itself.



Select the correlation to extract data from:
MeanEqualTimeGreensFunction
UpEqualTimeGreensFunction
DownEqualTimeGreensFunction
Density-densityCorrelationFn-UpUp
Density-densityCorrelationFn-UpDn
XXSpinCorrelationFunction
ZZSpinCorrelationFunction

3.1.4 Help

This menu is used for maintaining the version of ADEPT and linking to this user's guide.

3.1.4.1 Version

Opens a window showing the version of ADEPT.

3.2 File Lists

The file list panel displays one of the two maintained file lists in ADEPT; the user may toggle which list is displayed in the panel.

- Full file list: The entire list of files which have been added to ADEPT since execution. These files can be removed from the list by selecting them and hitting the `DELETE` key. `Shift` and `Ctrl` can also be used to select multiple files for removal.

- Filtered file list: A subset of the full file list and includes only the files which pass each filter. Therefore, if two filters were in place, only files which pass both filters would be placed in the filtered file list. ADEPT may allow for more logical operators (besides `AND`) in the future.

3.3 Filter Manager

Filters may be added to create a list of files for which each has values within some range set by the user. These values could be something like $Beta > 1.0$. Each filter has three parts, a variable, an operator, and a constant.

- Variable: Any variable which can be used in the data extraction variable lists is available to use in a filter. The value must be able to be parsed as a `double`.

- Operator: Currently, `double` comparison is used for filters, so operators available are: `<, <=, =, >=, >,` and $\neq$.
- Constant: A user-defined numerical value. This value is parsed as a double, but typing as an integer works as well.

Currently, only the logical `AND` operator is usable with filters, so a file will only be put into the *filtered file list* if it passes through all filters. This may be expanded to allow for more logical operators in future versions.

3.4 Variables

Variables are the values derived either directly from or through some manipulation of data found in the input files.

3.4.1 Extended variable list

The extended variable list is created upon each execution, but is invisible by default when another variable list is defined. This is because the extended variable list contains in it all data element types found in all input files. The purpose of such a list is to allow the user to extract any data element from any file if they so wish, even if that element is not defined in a user-defined variable list. The extended variable menu is split into three submenus:

- Parameters: Contains all data element types found in the first branch of the input data files, which are conventionally reserved for the parameters of the QUEST simulations. This would contain elements such as U, $Beta$, etc.
- Correlations: Contains submenus for each correlation defined in the input data files. A correlation is defined as an element which has children with the string "Point". These children must contain attributes for `Dx` and `Dy`.
- Non-Correlations: Contains all data elements which are not in the Parameters or Correlations menus.

3.4.2 User-defined variable lists

The Extended variable list is created by default by ADEPT, but the user may also define their own variable lists, which will appear as menus when choosing data elements from the Variables menus. These lists are read from an XML file passed in at the command line and are added upon execution to the menus. A checkbox will also be added to the Options menu if the user defines the list to have its visibility toggle-able. Below is an example variable list file, which will be explained after:

```
<?xml version="1.0" encoding="UTF-8"?>
<variablelists>
    <list name="Basic" settable="yes" defaultOn="yes">
        <list name="Basic Parameters">
            <variable name="t" description="Hopping amplitude" variable="T_up"/>
            <variable name="U" description="Interaction" variable="U"/>
            <variable name="mu" description="Chemical potential" variable="Mu_up"/>
            <variable name="beta" description="Inverse temperature" variable="Beta"/>
            <variable name="T" desctiption="Temperature" equation="1/Beta"/>
        </list>
        <variable name="Kinetic Energy" description="" variable="KineticEnergy"/>
        <variable name="Total Energy" description="" variable="TotalEnergy"/>
        <variable name="Density" description="" variable="Density"/>
    </list>
</variablelists>
```

For those not familiar with XML, the structure is fairly simply. It relies on nested lists of data elements. Every XML file has one overarching parent element, in this case `variablelists`. The top encoding line must be included at the beginning of each XML file which tells the parser how to read the file. The `variablelists` element can contain any number of the `list` element type. A `list` may contain any number `list` elements and `variable` elements (in any order).

List elements must contain three attributes:

- `name`: The name of the menu which will hold this list's variables/sub-lists
- `settable`: This option must only be set to either "yes" or "no" and indicates whether the menu's visibility can be toggled by the user during execution. For example, the Extended variable list is settable as seen in the Options menu.
- `defaultOn`: This option must only be set to either "yes" or "no" and indicates whether the menu should be visible upon execution or not. For example, the Extended variable list is visible by default when no other variable list is defined.

Variable elements must contain three attributes:

- `name`: The name of the variable
- `description`: This option defines a string which will be displayed when the user mouses over this variable in the menus.
- `variable/equation`: A variable element must contain either a `variable` or `equation` attrbitute.
  - `variable`: A variable attribute is used to make a reference to a variable which has already been defined. This can be used to avoid having to rewrite equations for multiple instances of the same variable.
  - `equation`: An equation defines a new variable based on previously defined variables - either directly from data files or from previous equation variables. For information on how to construct a valid equation, see 4.

3.4.3 Custom variables

The Custom menu allows the user to create new variables on-the-fly based on equations drawing on data from the input files. These variables will be saved and placed in the Custom menu above the "New" menu item. For information on creating a custom variable, see 4.

3.5 Result Panel

This panel displays all result values based on either the variables selected in the variable panel or the special outputs created by routines in the Output menu. The panel will often-times automatically update when a change is made which would affect its contents, but the panel can also be updated manually by pressing the **Update** button above the panel.

Rows may also be sorted based on values in a particular column by clicking the header for that column. Clicking the header again will sort the rows in the reverse direction.

4 Custom Variables

Custom variables represent equations which can pull any number of data elements from input files and manipulate their numerical values. For example, one could define `T = 1/beta`. This is a convenient way to express new variables to save to an output file and eventually plot. When the user defines a custom variable, either upon execution through a variable list input file or during execution via the Custom menu in the variable choice menus, it is created with a name and a String representation of the equation. The equation must contain at least one operand, either a constant or previously-defined variable, and can contain additional operands and operators.

Note: Since some variables are case sensitive, the user must enter the exact String being referenced. To aid this, auto-complete has been implemented which will try to complete the variable being typed by the user with a variable name which has already been defined.

4.1 Operands

Operands include previously-defined variables and constants, either written as integers or doubles. Constants will be parsed as doubles even if written as integers, so integer division is not possible. Points from correlations are accessed by the format: CorrelationName[X,Y]. Currently, the String for X and Y must match exactly to the coordinates found in the input data files, including 0-digits. The String is parsed literally; coordinates are not parsed as doubles.

4.2 Operators

Traditional mathematical operators are available, including `+, -, *, /, (, )`. Order of operations is maintained. The exponent symbol `^` is not included, but the function `pow` is available.

4.3 Functions

ADEPT uses developer-defined functions in the form `func(param1,param2,...)`. Current functions include:

- `abs(x)`: Returns the absolute value of `x`

- `avgcorr(correlationName)`: Returns the average value of points from the input correlation. Multiplicities are maintained based on the square lattice layout where the (0,0) counts once, each (x, 0) and (x, x) count four times, and all other points count eight times.
- `pow(base, exp)`: Returns the `base` to the power of `exp`, i.e. $base^{exp}$.
- `sqrt(x)`: Returns the square root of `x`.

5 Tutorials

Tutorials are available on YouTube which cover all topics described above and will be updated as features are added. These are available at `youtube.com/user/questadept`.

6 References

[1] `http://quest.ucdavis.edu/index.html`