User Manual for QU
v. 1.0

Pietro Longhi
pietro.longhi@gmail.com

October 11, 2016

Abstract

A short manual for using the software QU to produce generating series of 2d-4d framed BPS degeneracies with spin, to compute BPS monodromies from graphs.

1 Prerequisites

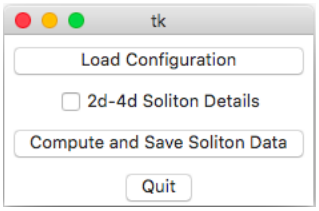
Running QU requires python 2.7, which can be downloaded for Windows, Mac OS and Linux at this link: `https://www.python.org/downloads/release/python-278/`. Note that Linux and Moc OS likely often existing python distributions, which may be sufficient for running QU.

2 Graphical interface

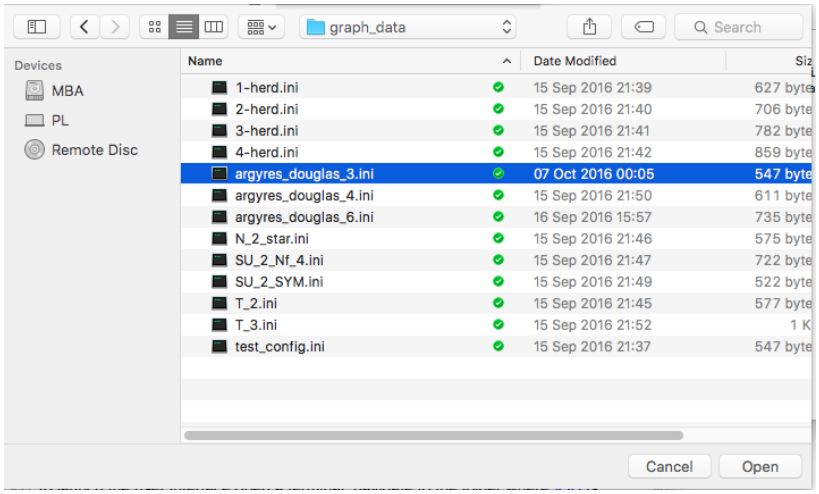
QU comes with an elementary graphical user interface, providing the most basic functionalities. To launch the user interface open a terminal, navigate to the folder where QU is located, and execute the command

```
$ python gui.py
```

This will open a window like the one shown in figure below



Clicking on the button Load configuration will open a window which allows to select a configuration file, containing topological data of the graph (we explain how to prepare such a file in the next section, some examples are provided in the sub-folder graph_data).



Select a configuration file, for example argyres_douglas_3.ini, and click Open. This will bring back the former window, now click on Compute and Save Soliton Data, and select a place to save the data that will be produced, for example argyres_douglas_3.txt.

QU will now compute generating functions $Q^{\pm}(p, y)$ defined in the paper "Wall crossing invariants from Spectral Networks" by the author. The output will be printed in the file specified in the last step above, in our example this is the file argyres_douglas_3.txt in the folder results. If the configuration file of the network is formatted correctly, a message similar to the following message will be printed at the top of the output file:

```
All streets have two well-defined endpoints.
All joints are of a well-defined type.
All homology classes are well-defined.
```

Shortly below this message a short dictionary between variables and homology classes will be printed:

```
Dictionary of formal variables and homology basis:
gamma_1 ---> X1
gamma_2 ---> X2
```

The data will be contained under the section SOLITON DATA, for example

```
===============
SOLITON DATA
===============

Data of street p_2
In american resolution
---------------------------

4d Soliton generating function (with spin)
X2/y + 1

Data of street p_1
In american resolution
---------------------------

4d Soliton generating function (with spin)
X1*X2/y + X1/y + 1
```

et cetera. The data should be read according to the dictionary printed above. For example, in the language of the paper "Wall crossing invariants from spectral networks", this means that

$$\begin{aligned} Q^{(+)}(p_1, y) &= 1 + y^{-1}\,\hat{Y}_{\tilde{\gamma}_1} + y^{-1}\hat{Y}_{\tilde{\gamma}_1+\tilde{\gamma}_2}\,, \\ Q^{(+)}(p_2, y) &= 1 + y^{-1}\,\hat{Y}_{\tilde{\gamma}_2}\,. \end{aligned} \tag{1}$$

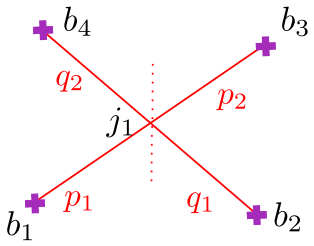
Note an important point on conventions: the monomial `X1*X2` does *not* correspond to $\hat{Y}_{\tilde{\gamma}_1}\hat{Y}_{\tilde{\gamma}_2}$, but to $\hat{Y}_{\tilde{\gamma}_1+\tilde{\gamma}_2}$!

3 Constructing a graph

The critical graph $\mathcal{W}_c$ is given to QU by writing a text file which must be saved with the .ini extension. Several templates are included in the folder graph_data. The basic structure of the file includes two sections: Network Data and Computation Parameters. For example, the file 1-herd.ini reads

```
[Network Data]

streets = ['p_1','p_2','q_1','q_2']
branch_points = {
   'b_1' :  ['p_1'],
   'b_2' :  ['q_1'],
   'b_3' :  ['p_2'],
   'b_4' :  ['q_2']}

joints = {
   'j_1':  ['p_1', None, 'q_1', 'p_2', None, 'q_2' ]}

homology_classes = {
   'gamma_1' :  ['p_1', 'p_2'],
   'gamma_2' :  ['q_1', 'q_2'],}

[Computation Parameters]

iterations = 10
```

This corresponds to the graph shown in figure below



- In streets one must specify a list of labels for streets, the list is delimited by *square brackets*, and each name is delimited by single quotation marks.
- Then one specifies a list of branch points: this is delimited by *curly brackets*, each entry is of the form

  `'label' :  ['street_1', 'street_2', 'street_3'],`

where the first entry is a label for the branch point, the separator is a *colon*, and the second entry is a list in *square brackets*, containing up to three labels of edges which end on the branch point. The ordering from left to right corresponds to going clockwise around the branch point.

- Then one specifies joints, the structure is identical to that of branch points, except that the labels of edges ending on a joint are contained in a list of up to six elements. Important: if there is an empty slot between two edges anding at the joint, this must be specified! In the empty slot one must write `None`, see the example given above.

- The last field is that of `homology_classes`. As the name suggests, this tells QU which streets lift to form a homology class. In the example, the lift of streets p_1 and p_2 corresponds to a cycle in homology class γ_1. This is specified by adding to the list the entry

  ```
  'gamma_1' :  ['p_1', 'p_2'],
  ```

  with the labels of the two streets p_1 and p_2. Again it is important to note that the overall delimiters of the list are curly brackets, while the delimiters of the sub-list of street labels are square brackets, the separator is a colon.

- Finally, in the section Computation Parameters one specifies how many iterations of equation solving QU should attempt to do, by giving a positive integer.