Kei_Hm_Finder Documentation

Usage Guide and Reference Manual

By Tongchen He
5/20/2022

Table of Contents

Introduction:

Purpose:

In addition to the Wirt_Hm_Suite_Sage package developed by N. Morrison, Kei_Hm_Finder searches for surjective homomorphisms from the fundamental kei of a topological knot with Wirtinger number n to a finite kei with rank n. The existence of such a surjective homomorphism further proves the Cappell-Shaneson Meridional Rank Conjecture for the knot.

This program uses the calc_wirt algorithm developed by P. Villanueva and the main structure of gauss_processor, excel_reader.py and kei_hm.py are developed by N. Morrison. Details of these modules can be found in *Wirt_Hm_Suite_Sage_Documentation*, by Nathaniel Morrison.

Requirements to run:

1. Python 3 or later.
2. A Microsoft Excel installation. Excel 2010 or later is recommended.
3. The following python packages need to be installed in the python interpreter:
    1. XlsxWriter
    2. numpy
    3. openpyxl
4. The following files show be placed in the same directory:
    1. Calc_wirt.py
    2. excel_reader.py
    3. gauss_processor.py
    4. kei_database.xlsx[1]
    5. kei_hm.py
    6. kei_main.py

General Usage

Run kei_main.py on any python IDE, the program will then start and present a menu. Explanations for menu options:

1. **Menu 1:** This menu pops up at the start of the program: "Enter 1 if you would like to input the Gauss code of a single knot, 2 if you would like to input an Excel file"

[1] The data in this file is obtained from http://shell.cas.usf.edu/~saito/QuandleColor/ by W. Edwin Clark and Timothy Yeatman.

   a. **Option 1-1:** User chooses to input the Gauss code for a single knot, then the second menu prompt will appear (check Menu 2 for details).
      i. **Input explanation:** Accepts as input a list of positive and negative integers giving the Gauss code of the knot. The integers must be separated by non-numeric characters, but beyond this the choice of separation characters and the presence or absence of non-numeric leading or trailing characters is irrelevant. Example valid gauss codes: 1 -2 3 -4 5 -6 2 -1 6 -3 4 -5 ; {1,-2,3,-4,5,-6,2,-1,6,-3,4,-5} ; put1w-2ha3t-4ever5y-6o2u-1w6a-3nt4h-5ere!
   b. **Option 1-2:** User chooses to input an excel file containing knot data. The program will then ask the user to input the directory of the excel file. It will search for surjective homomorphisms from every knot in the excel file to every kei in the kei database (kei_database.xlsx).
      i. **Input explanation:** An example of the file directory is C:\Users\Me\InputFiles\knotList.xlsx. Ensure that the name and Gauss code of each knot is present in the file, that all names are placed in one column and all codes are placed in a different column, and that there are no empty rows between the first row of knot data and the last row of knot data. The program will also ask if the user has the data of wirtinger number of each knot, but this is only optional.

2. **Menu 2:** If the user chooses option 1 from the previous selection, then the following prompt will pop up: "Enter 1 if you would like to test the knot on a single kei, 2 if you would like to test them on all finite keis under order 36: "
   a. **Option 2-1:** User chooses to test the knots on a single kei. The program will then ask the user for the RIG index of the knot and search for surjective homomorphism from the single knot to the single kei.
      i. **Input explanation:** The program only accepts input format exactly like the following: RIG[1]. Users can look up kei_database.xlsx for the RIG index of the desired kei under order 36.
   b. **Option 2-2:** User chooses to test the knots on every kei under order 36. The program will then search for surjective homomorphism from the single knot to every kei under order 36. No further input is needed.

Module Documentation:

Modules: calc_wirt.py, gauss_processor.py

For details of these two modules, refer to Nathaniel Morrison, *Wirt_Hm_Suite_Sage_Documentation*, pg. 8, 10-11.

Module: excel_reader.py

Functions:

1. kei_processor(raw_kei): converts a string of kei information into a 2D list of a kei table
    a. input:
        i. raw_kei: a string containing the multiplication table of a kei, needs to contain a perfect square number of integers, any separation is accepted.
    b. output
        i. kei: a 2D list containing the multiplication of the kei
2. kei_dict_creator(file_name): retrieves the information of keis, including their names, ranks, and multiplication tables from an excel file, and store them in a dictionary format
    a. input:
        i. file_name: a string containing the directory and name of the excel file where all the kei information is stored
    b. output:
        i. kei_dict: a dictionary where the keys are names (RIG index) of keis, and values as two-tuples whose elements are the corresponding rank and multiplication table of each kei
3. excel_main(input_name): Prompts the user to input parameters for input excel file, retrieves knot information from the input file and kei information from kei_database.xlsx, calls other modules to look for homomorphism from each knot to each kei, and finally insert those data into an output excel file
    a. input:
        i. input_name: a string containing the name and directory of the input file
    b. output: None (the function places the output data in a new excel file instead of any return value)

For functions knot_processor, excel_creator, and excel_writer, refer to Nathaniel Morrison, *Wirt_Hm_Suite_Sage_Documentation*, pg. 9-10. The only difference is that this module also stores mapping information in the output file.

Module: kei_hm.py

Functions:

1. generator_finder(rank, q1): given the rank of a kei, find all of its minimal generating sets
    a. Input:
        i. rank: the rank of the kei, i.e. minimal number of elements to generate the whole kei
        ii. q1: the kei being worked on
    b. Output:
        i. list_of_gen: A list containing tuples of every minimal generating set of the kei

For functions homomorphism_finder, strand_assignment, generator_assign, and hom_tester, refer to Nathaniel Morrison, *Wirt_Hm_Suite_Sage_Documentation*, pg. 14. The main idea is similar, but the difference is that the functions in kei_hm take keis as parameters and output elements/homomorphisms related to keis.

Module: kei_main.py

Functions:

1. find_hm(kei_index, kei, rank, gcode): Calls other modules to search for homomorphisms from the given knot to the given kei, then prints out the result
    a. input:
        i. kei_index: the RIG index of the kei
        ii. kei: A 2D list containing the multiplication table of the kei
        iii. rank: the rank of the kei, i.e. minimal number of elements to generate the kei
    b. output: None
2. kei_main(): main function of the module, prompts the user to enter input, operates on other modules, and displays results. Usage guide can be found here: General Usage.
    a. input: None
    b. output: None