Python tools for working with matrix groups
Development roadmap

Joshua Ruiter

December 19, 2025

Contents

What's next

Automate extraction of roots from the torus conjugation on the Lie algebra. Probably going to need a "generic Lie algebra element" map first.

Last updated: Dec 19, 2025

1 Overall goals

The goal of this project is to have a self-contained and comprehensive set of tools for performing calculations and verifying properties of matrix groups, especially special linear, special orthogonal, and special unitary groups. There are two main tasks I want the code to be able to perform:

1. Given some basic information about a matrix group (such as the equational condition for membership, a torus subset, and the equational condition for membership in the Lie algebra), perform calculations to determine a lot of information such as:
    - Root system/root datum, root spaces (and their dimensions) inside the Lie algebra
    - Root subgroups, pinning maps, and homomorphism defect coefficients
    - Summable pairs of roots, commutator coefficients
    - Weyl group elements, Weyl group conjugation coefficients,
2. Verify that all of the information above satisfies the required relations.

In order to facilitate the above tasks, particularly those related to root systems, it will also be necessary to have a class to model root systems (or perhaps root data), and perform certain calculations such as enumerating summable pairs of roots.

2 Code documentation

2.1 Root system class

Current status: Effectively complete. I have a working class which implements root systems effectively. It stores internal variables of dynkin_type, rank, vector_length, name_string, and root_list. The main interesting data is the root_list, which is a list of vectors.

Potential future tasks: There are a few small things which would be nice to eventually implement, but only after a lot of other things are complete. These are:

- Implement root list builders for types D, E, F, and G. This should not be very difficult.
- Implement an alternative constructor which allows a user to build a root system out of their own list of vectors, rather than using prepackaged root lists. This would allow for more generality of use of the class.
- Implement some kind of internal model or calculation tool for the Weyl group of a root system. Not exactly sure if this is necessary, but it could be useful at some point, who knows.
- Generalize by replacing root system with root datum. A root datum stores slightly more information that a root system, and sometimes this level of generality is necessary for working with algebraic groups. I am skeptical this will be necessary for anything I want to do, so I will wait until this becomes absolutely necessary.

2.2 Nondegenerate isotropic form class

Current status: Effectively complete. I suspect that "nondegenerate isotropic form" is unnecessarily specific, and I should have a "bilinear form" or something, but this is doing the job it needs to right now.

Potential future tasks: Rewrite the entire thing at a better level of generality? This will become apparent if it needs to be done, and otherwise nothing needs to be done.

2.3 Pinned group class

Current status: I have a pinned_group class, but it isn't set up in the way that I want so I think I will rebuild it, utilizing some of the existing code but I'm not sure how much. I envision using in the following way:

- Initialize a pinned group object by giving it only basic data. Hopefully this is restricted to just the name, matrix size, bilinear form (if there is one), condition for group membership, condition for torus membership, and a map to output a generic element of the torus. Example code:

```
sl3 = pinned_group(name_string = "special linear",
                   matrix_size = 3,
                   form = None,
                   is_group_element = lambda X : shape(X) == (3,3) and det(X) == 1,
                   is_torus_element = is_diagonal,
                   generic_torus_element = lambda [t1,t2,t3] : diag(t1, t2, t3))
```

- After initializing a pinned group object as above, use a single line of code to invoke a large number of calculations to determine all of the following, if possible: Lie algebra membership condition, root system, root spaces, map to output a generic element of a root space, root subgroups, map to output a generic element of a root subgroup, homomorphism defect coefficients, commutator coefficients, Weyl group elements, Weyl group coefficients. Example code (to run immediately after the above block):

  ```
  sl3.fit_pinning(display = True)
  ```

  When run with the display=True setting, this should output a variety of visual representations of the calculated information. Regardless of the display settings, it should change various internal aspects of the sl3 pinned_group object so that it is now capable of being used for various calculations.

- In particular, after a .fit() method is run, it should be possible to run something like

  ```
  sl3.verify_pinning()
  ```

  which runs a variety of tests to check all the formulas related to pinnings, i.e. torus conjugation formula, pseudo-homomorphism property, commutator formula, Weyl group conjugation formula.

3 Theoretical background

3.1 Fitting pinnings