ADG - Automated Diagram Generator Documentation

Release 3.0.3

ADG Dev Team

Mar 20, 2023

Contents

CHAPTER 1

The ADG Project

1.1 Description

ADG is a tool generating diagrams and producing their expressions for given many-body formalisms. Diagrammatic rules from the formalism are combined with graph theory objects to produce diagrams and expressions in a fast, simple and error-safe way.

The only input consists in the theory and order of interest, and the N-body character of the operators of interest. The main output is a LaTeX file containing the diagrams, their associated expressions and additional informations that can be compiled into PDF by ADG if needed. Other computer-readable files may be produced as well.

1.2 Status

As for now, the code is capable of handling four different formalisms, i.e. Many-Body Perturbation Theory (MBPT), Bogoliubov Many-Body Perturbation Theory (BMBPT), Projected Bogoliubov Many-Body Perturbation Theory (PBMBPT), and Bogoliubov In-Medium Similarity Renormalization Group (BIMSRG).

- For MBPT, the code generates all Hartree-Fock energy diagrams at any given order along with their expression and additional information (conjugate diagram, excitation level...).
- For (P)BMBPT, the code generates all diagrams for a generic observable commuting with the Hamiltonian, along with their time-dependent and time-integrated expressions.
- For BIMSRG, the code generates all diagrams and expressions at any given truncation order for the two operators as well as the commutator itself. The traditional BIMSRG(n) truncation order corresponds to truncating both operators as well as the commutator at the same rank.

1.3 Future developments

ADG is currently being extended to diagrams and expressions generation for Gorkov Self-Consistent Green's Functions (GSCGF).

CHAPTER 2

Install ADG on your computer

2.1 Install

To install ADG, download the source files and run

```
pip install <project_folder>
```

or alternatively

```
python setup.py install
```

If you want to install ADG in `develop` mode, then run

```
pip install -e <project_folder>
```

2.2 Dependencies

In order to run the code, you will need a Python2 install >= 2.7.1 and the following Python libraries:

- networkx >= 2.0
- numpy
- scipy
- future
- more-itertools

If you want ADG to compile the LaTeX output file, you will need a Latex install with the PDFLaTeX compiler and the feynmp and feynmp-auto packages installed, which are standard packages in most recent distributions.

CHAPTER 3

Generate diagrams with ADG

3.1 Run ADG

To run the program and generate BMBPT diagrams at order 4 for example, use

```
adg -o 4 -t BMBPT -d -c
```

where the `-o` flag is for the order, `-t` for the type of theory, `-d` indicates you want the diagrams to be drawn and `-c` that you want ADG to compile the LaTeX output.

You can alternatively run the program in interactive mode by typing

```
adg -i
```

Finally, to obtain more information on all the available flags, use

```
adg -h
```

3.2 CLI options

3.2.1 Generic options:

-o, --order	order of the diagrams [1-9] or N_A, N_B, N_C truncation for BIMSRG
-t, --theory	theory of interest: MBPT, BMBPT or PBMBPT
-i, --interactive	execute ADG in interactive mode

3.2.2 (P)BMBPT options:

-can, --canonical	consider only canonical diagrams
-nobs, --nbody_observable	maximal n-body character of the observable [1-3], default = 2
-3NF, --with_3NF	use two and three-body forces for BMBPT diagrams
-dt, --draw_tsds	draw Time-Structure Diagrams

3.2.3 MBPT option:

-cd, --cd_output produce computer-readable output for automated frameworks

3.2.4 Run management options:

-d, --draw_diags draw the diagrams using FeynMF

-c, --compile compile the LaTeX output file with PDFLaTeX

3.3 Output files

The output of the program is stored in a folder named after the theory, and a subfolder named after the order, e.g. `/MBPT/Order-4`. In the case of (P)BMBPT, suffixes are added depending on the n-body forces of the observable, and if three-body forces were used or only canonical diagrams computed, i.e. for our previous example, results would be stored under `BMBPT/Order-4_2body_observable`. For BIMSRG, the folder corresponds e.g. to `BIMSRG/Order_1_2_3` if N_A is 1, N_B is 2 and N_B is 3.

The main output file of the program, called `result.tex`, is a LaTeX file containing the expressions of the diagrams along other basic infos on their structure, and, if flag `-d` has been used, drawing instructions. The file is automatically compiled and produces a PDF file `result.pdf` when using the `-c` file.

A list of the adjacency matrices associated with the diagrams is printed separately in the `adj_matrices.list` file to allow for an easy use with another many-body diagrams code.

In the case of a MBPT calculations, it is possible to produce output specifically tailored for automated calculations framework by using the `-cd` flag. The associated output files use `CD_` as a prefix.

CHAPTER 4

ADG Reference for Developers

4.1 Main script

4.2 Run & CLI management

4.3 Generic Diagram

4.4 MBPT diagram

4.5 BMBPT Diagram

4.6 PBMBPT Diagram

4.7 Time-Structure Diagram

4.8 BIMSRG Diagram

4.9 Tools and utilities

Miscellaneous diagram-unrelated tools for ADG.

class adg.tools.**UniqueID**

Bases: object

Counter making sure of generating a unique ID number for diagrams.

current

The unique identifier to be attributedto a diagram.

Type int

current

get()
Iterate on counter value and return current value.

Returns A unique identifier for the diagram.

Return type (int)

adg.tools.**reversed_enumerate**(*data*)
Return the index and item of the data in reversed order.

Parameters **data** (*iterable data structure*) – The data to be used..

Returns Index and item.

Return type (tuple)

```
>>> list(reversed_enumerate(['A', 'B', 'C']))
[(2, 'C'), (1, 'B'), (0, 'A')]
```

CHAPTER 5

Developers Team

They have been involved in the making of ADG over the past years:

- Pierre Arthuis - TU Darmstadt & ExtreMe Matter Institute EMMI, GSI, Darmstadt (previously University of Surrey & Irfu, CEA, Université Paris-Saclay & CEA, DAM, DIF)
- Thomas Duguet - Irfu, CEA, Université Paris-Saclay & KU Leuven, IKS
- Jean-Paul Ebran - CEA, DAM, DIF
- Heiko Hergert - NSCL/FRIB Laboratory & Department of Physics and Astronomy, Michigan State University
- Raphaël-David Lasseri - ESNT, Irfu, CEA, Université Paris-Saclay (previously IPN, CNRS/IN2P3, Université Paris-Sud, Université Paris-Saclay)
- Julien Ripoche - CEA, DAM, DIF
- Alexander Tichai - MPI fuer Kernphysik, Heidelberg & TU Darmstadt & ExtreMe Matter Institute EMMI, GSI, Darmstadt (previsously ESNT, Irfu, CEA, Université Paris-Saclay)

CHAPTER 6

Citing

If you use ADG in your research work, we kindly ask you to cite the following papers:

- P. Arthuis, T. Duguet, A. Tichai, R.-D. Lasseri and J.-P. Ebran, Comput. Phys. Commun. **240**, 202-227 (2019). It is available under the following DOI.
- P. Arthuis, A. Tichai, J. Ripoche and T. Duguet, Comput. Phys. Commun. **261**, 107677 (2021). It is available here.
- A. Tichai, P. Arthuis, H. Hergert and T. Duguet, Eur. Phys. J. A **58**, 2 (2022). It is available under this URL.

CHAPTER 7

License

ADG is licensed under under GNU General Public License version 3 (see LICENSE.txt for the full GPLv3 License).

```
Copyright (C) 2018-2023 ADG Dev Team
Pierre Arthuis
Thomas Duguet
Jean-Paul Ebran
Heiko Hergert
Raphaël-David Lasseri
Julien Ripoche
Alexander Tichai
```

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index