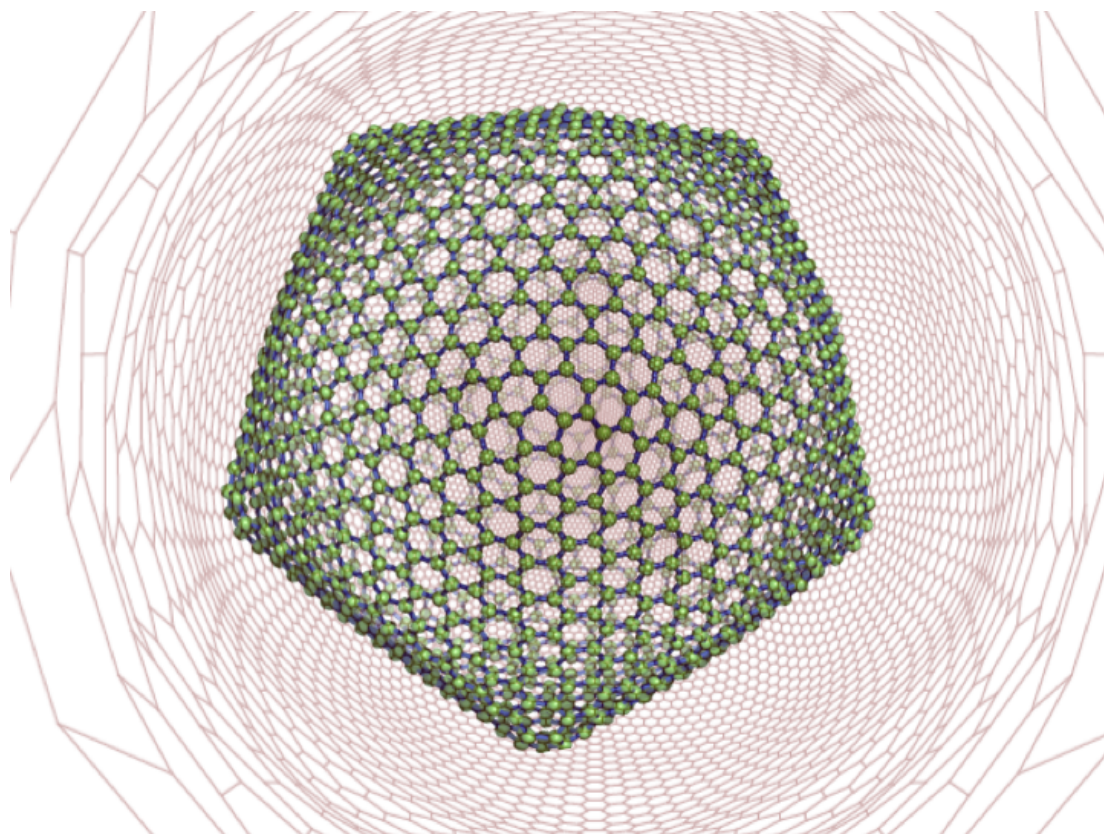


Fullerene (Version 4.5)
A Program for the Topological Analysis of Fullerenes
User's Manual



Peter Schwerdtfeger* and Lukas Wirz†

Centre of Theoretical Chemistry and Physics, The New Zealand Institute for Advanced Study,
Massey University Auckland, Private Bag 102904, North Shore City,
0745 Auckland, New Zealand.

James Avery‡

Niels Bohr Institute, University of Copenhagen, 2100 Copenhagen, Denmark.

October 20, 2015

*Email: p.a.schwerdtfeger@massey.ac.nz

†Email: l.wirz@massey.ac.nz

‡Email: avery@nbi.ku.dk

Important Copyright Message:

This is an **open-source code** and you may use and modify the program at your own will. If you like to know why, read the article by Darrel C. Ince, Leslie Hatton, John Graham-Cumming, *Nature* **482**, p.485 (2012). However, we kindly ask that if our program is used, and data are subsequently published, to cite the references given below. Before you distribute the program to other agencies or users outside your group/department please make them aware that their name should be added into our user's database. For more details go to our website at Massey University:

<http://ctcp.massey.ac.nz/index.php?group=&page=fullerenes&menu=fullerenes>

For any questions concerning this program please feel free to contact us. We are always open to improvements and suggestions.

Please cite the following paper if you use this program for publishing data:

1. P. Schwerdtfeger, L. Wirz, J. Avery, *Fullerene – A Software Package for Constructing and Analyzing Structures of Regular Fullerenes*, J. Comput. Chem. **34**, 1508–1526 (2013).

We also recommend to cite the following book (many of the concepts used can be found here):

2. P. W. Fowler and D. E. Manolopoulos, *An Atlas of Fullerenes* (Dover Publ., New York, 2006). This book is highly recommended. It helps understanding how this program functions.

Further suggested reading:

3. P. Schwerdtfeger, L. Wirz, J. Avery, *The Topology of Fullerenes*, WIRE Comput. Mol. Sci. **5**, 96–145 (2015).
4. L. N. Wirz, R. Sure, R. Tonner, A. Hermann, P. Schwerdtfeger, *A Harmonic Force-Field Method for Fullerenes and a Comparison to Density Functional Calculations for Goldberg-Coxeter Fullerenes up to C₉₈₀*, J. Comput. Chem., in press.
5. D. Babić, *Nomenclature and Coding of Fullerenes*, J. Chem. Inf. Comput. Sci. **35**, 515–526 (1995).
6. D. E. Manolopoulos and P. W. Fowler, *Molecular graphs, point groups, and fullerenes*, J. Chem. Phys. **96**, 7603–7614 (1992).
7. Z. C. Wu, D. A. Jelski, and T. F. George, *Vibrational Motions of Buckminsterfullerene*, Chem. Phys. Lett. **137**, 291–295 (1987).
8. G. B. Adams, M. O’Keefe, and R. S. Ruoff, *Van der Waals Surface Areas and Volumes of Fullerenes*, J. Phys. Chem. **98**, 9465–9469 (1994).
9. G. Brinkmann, P. W. Fowler, *A Catalogue of Growth Transformations of Fullerene Polyhedra*, J. Chem. Inf. Comput. Sci. **43**, 1837–1843 (2003).
10. D. Babić, D. J. Klein and C. H. Sah, *Symmetry of fullerenes*, Chem. Phys. Lett. **211**, 235–241 (1993).
11. T. Pisanski, B. Plestenjak, and A. Graovac, *NiceGraph Program and its applications in chemistry*, Croatica Chemica Acta **68**, 283–292 (1995).
12. B. Plestenjak, *An algorithm for drawing Schlegel diagrams*, <http://www-lp.fmf.uni-lj.si/plestenjak/Papers/NICEGR.pdf>.
13. J. Bondy and U. Murty, *Graph Theory* (Springer, Berlin, 2008).
14. A. J. M. Wilson, *Graphs and Applications. An Introductory Approach* (Springer, Berlin, 2000).
15. J. Cioslowski, N. Rao, and D. Moncrieff, *Standard Enthalpies of Formation of Fullerenes and Their Dependence on Structural Motifs*, J. Am. Chem. Soc. **122**, 8265–8270 (2000).
16. M. Alcami, G. Sanchez, S. Diaz-Tendero, Y. Wang, F. Martin, *Structural Patterns in Fullerenes Showing Adjacent Pentagons: C₂₀ to C₇₂*, J. Nanosci. Nanotech. **7**, 1329–1338 (2007).
17. F. Cataldo, A. Graovac, O. Ori *The Mathematics and Topology of Fullerenes* (Springer, Berlin, 2011).

Contents

Contents	4
1 Introduction	5
2 Installation and running the program	7
3 Program Structure	8
4 Input description	19
5 Fullerene Isomer Database	27
Bibliography	28

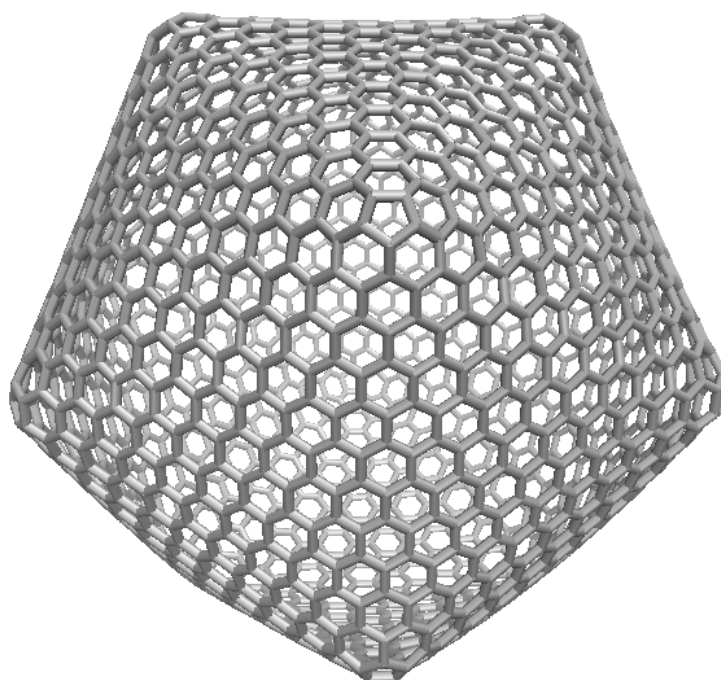


Figure 1: Force-field optimized structure of I_h -C₂₁₆₀ viewed with VMD [1].

Acknowledgement: *PS is indebted to the Alexander von Humboldt Foundation (Bonn) for financial support in terms of a Humboldt Research Award, and to both Prof. Gernot Frenking and Dr. Ralf Tonner (Marburg) for support during his extended stay in Marburg where writing of this program began. We acknowledge the help of Darko Babić, Patrick W. Fowler (Sheffield) and David E. Manolopoulos (Oxford) for permission to freely distribute their Fortran subroutines which we modified and implemented in this program package. We also thank Prof. Ottorino Ori (Zagreb) and Jan Goedgebeur (Gent) for fruitful discussions.*

1 Introduction

Fullerene is a general purpose computer program that creates the adjacency matrix for any fullerene isomer, performs a topological/graph theoretical analysis, and creates a reasonably accurate 3D structure (cartesian or internal coordinates) through various different methods and algorithms. The results can be used for plotting 2D fullerene graphs (e.g., Schlegel diagrams) and 3D structures, and serves as a good starting point for further quantum theoretical treatment. The program is written in standard Fortran and C++ and is easily implemented within a Linux/UNIX environment. Although there are several other computer programs already available to deal with fullerene graphs like *CaGe(plantri)*, [2] *fullgen*, [3] *Buckygen* [4, 5], [6] *Vega*, [7], *FuiGui*, [8] or the routines introduced in the book by Fowler and Manolopoulos, [9] we felt that there is a need to write a general purpose program which does what other programs do and (of course) a lot more. The first version (1.0) was written in Fortran for calculating the surface area and volume of a fullerene and corresponding changes due to endohedral incorporation of rare gas atoms [10]. A much improved version 2.0 allowed for the creation of fullerene structures using the ring-spiral algorithm introduced in the book by Fowler and Manolopoulos [9], and version 3.0 was already dealing with topological and graph theoretical properties. It became soon clear that many open problems remain in fullerene structure generations which needed to be addressed, [11] and it was therefore decided to extend version 3 to a general purpose program package and make it available to the scientific community. [12]

In the current version only regular fullerenes are considered (i.e. of genus 0 consisting of pentagons and hexagons only) fulfilling Euler's theorem, $n_v - n_e + n_f = 2$, where n_v is the number of vertices (atoms), n_e the number of edges (bonds), and n_f the number of faces (5- or 6-rings). The program constructs an accurate 3D structure (in cartesian coordinates) of any fullerene isomer from the canonical ring spiral pentagon indices (RSPI) (or its generalized version) through either a 3D Tutte embedding (3D-TE) [13] or the adjacency matrix eigenvector (AME) method [9]. One can also construct the n -th leapfrog $GC_{1,1}^n[C_{n_v}]$ fullerene, the halma fullerene $GC_{k,0}[C_{n_v}]$ or the more general Goldberg-Coxeter transformation of a fullerene $GC_{k,l}$ with $k \geq l$, which is implemented not only for C_{20} , i.e. $GC_{k,l}[C_{20}]$, as in a previous version (4.4) of the program, but also for any fullerene using an algorithm developed by Avery. [14] Starting from a given fullerene structure one can perform vertex insertions (such as Endo-Kroto [15], Yoshida-Fowler [16] or Brinkmann-Fowler [17]) or Stone-Wales transformations [18]. Vertex insertions allow for the construction of non-spiral fullerenes such as C_{380} or C_{384} . Geometry optimization using a Fletcher-Reeves-Polak-Ribiere minimization [19] with analytical gradients for the harmonic oscillator force field [20, 21] is implemented in the current version, providing a good initial guess for 3D cartesian coordinates. 2D fullerene graphs (e.g. Schlegel diagrams) can be produced using a variety of different algorithms [22]. The program determines the volume and surface area of a fullerene (irregular or not) through tessellation in trigonal pyramids or from its convex hull, and gives measures for spherical distortion and convexity. It determines the minimum covering sphere, the minimum distance sphere and the maximum inner sphere. The program further calculates the number of Hamiltonian cycles and produces ring spiral indices. Cioslowski's scheme for the calculation of the heat of formation for IPR (isolated pentagon rule) fullerenes is implemented [23], as well as Babić's scheme for calculating the total resonance energy of a general fullerene [24, 25], and Martin's scheme for the heat of formation [26]. A variety of topological indicators, such as the Wiener index, Szeged index and Balaban index are calculated. [27, 28]

This program works for any (distorted or not) regular fullerene. The spiral algorithm of Fowler and Manolopoulos used here [9] is not restricted to a pentagon start or to canonical indices. For a general list of fullerenes see "The House of Graphs" [29]. The program produces an external file with default name **Fullerene-3D.xyz**, **Fullerene-3D.cc1** or **Fullerene-3D.mol** to be used for standard molecular plotting programs like CYLview [30], Avogadro [31], Jmol [32], Pymol [33] or VMD [1]. For using these programs it is important that the fullerene has a reasonable 3D structure, obtained for example by a force field optimization, otherwise bonds cannot be correctly identified from the .xyz file only. It may therefore not be advisable to use structures that come out directly from the AME or 3D-TE algorithm, or the .cc1 file could be used containing information on vertex adjacencies. We recommend VMD [1], it is more robust and works even for the largest fullerenes beyond 1000 atoms. The program produces fullerene 2D graphs in latex format, but can be used by any other program which is capable of producing 2D graphs, for example QMGA [34]. For this an external file is written out called **Fullerene-2D.dat** containing all the information required for plotting a 2D graph.

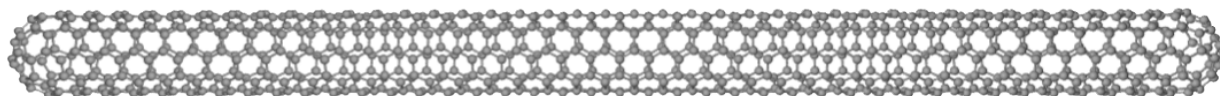
From version 4.0 onwards (released July 2012) C++ routines embedded into the original Fortran program incorporate much improved algorithms compared to the older versions. The reason for using two different program languages is that PS is good in old-fashioned Fortran, JA is good in C++ (and LW is doing

both). Furthermore, some of the original routines used were available in Fortran only. Some standard routines from Mathematical Recipes were modified for the purpose of matrix diagonalization and geometry optimization. Version 4.3 included the Hessian of the force field, and in this version (4.5) force constants are used which match experimental or calculated vibrational frequencies. Version 4.4 introduced a search algorithm for neighbourhood RSPIs, the Szeged index, pentagon arm indices as further topological descriptors, and new (conjectured) upper and lower bound for Hamiltonian cycles in fullerenes. Version 4.5 has a new algorithm for general Goldberg-Coxeter transforms [14]. The Tutte embedding algorithm has also been optimized to perform much faster, and the fullerene database has been transformed into a more compact and efficient version. Version 5 to be released late 2016 will separate the topological from the cartesian part and uses sparse matrix handling for the largest fullerenes. This program is under permanent construction and has been tested extensively for bugs. Nevertheless, if you have any problem or find any bugs please report to one of us.

There are many things on our to-do list. We try hard to release more functionality to this program in due time. Not implemented yet is:

- The minimum covering ellipsoid useful for rugby-ball shaped fullerenes, and close packing of ellipsoids;
- Symmetrization of optimized coordinates to point group symmetry (or a chosen lower symmetry) and symmetry labels for Hückel orbital energies and vibrational frequencies;
- Extension to general cubic polyhedra of genus 0 (heptagons and squares);
- Extension to non-regular fullerenes of higher genus;
- A restart option for subroutine Hamilton as it is an $\#P$ -hard problem and we like to test the limitations of the back-track algorithm used in program;
- Kekulé structures (perfect matchings, so far only the counting is implemented) and Clar sextets;
- General plotting program (other than latex) for producing Schlegel diagrams and corresponding duals with a good graphical interface similar to the *CaGe* program;[6]
- Further extension of the fullerene database;
- Implementation of the Brinkmann-Goedgebeur fast-generation algorithm for fullerenes, which is more efficient than the original ring-spiral code of Fowler and Manolopoulos.

We are also open to any other suggestions and welcome additions to this program suite. In this case please feel free to contact us per email.



2 Installation and running the program

All fortran source files are located in the directory “source” and all C++ files are in “libgraph”.

The program is Linux/UNIX based and is tested to compile with the gnu compiler collection (more specifically gfortran and g++). You need to use the **Makefile** included in the **fullerene.zip** file provided and type

```
make
```

In the Makefile several compile flags are set. Per default, the machine dependent flag **-m64** is set (if you are using a 32-bit environment you will need to change this to **-m32**). The optimization level is set to **-O3**. If you require an atom count above 5000 (the upper limit can be set in **source/config.f**, see entry **Nmax=5000**) there will be arrays which are larger than 2 GB. Per default most compilers don't support this. If your machine has enough memory, your gfortran is sufficiently new (4.4.3 to 4.4.7 and 4.6 on 64-bit Linux work) you can set **-mcmodel** to **medium** to switch to the medium code model in which objects are permitted to be larger than 2 GB (on Macs this may not always work). All binary objects must be compiled with the same options; delete all binaries before you recompile using a different code model.

The executable “fullerene” can be executed on a Linux/UNIX as

```
./fullerene < filename.inp > filename.out
```

A number of test input files can be found in the directory “input”. If you type

```
make tests
```

it will run all the input jobs and stores them into the output directory as *.out. If you type

```
make clean
```

all the object files are deleted. Additionally, the linked binaries are deleted if you type

```
make distclean
```

All input and output files provided have been checked for correctness. The test files will run a couple of minutes on a fast intel processor.

If you use our fullerene database (which is recommended for time savings), the directory **database** needs to be created (if not already there) and moved into the main directory where the **source** and **libgraph** directories are located. Please download our databases and move it into this folder. **database** should contain the folders **ALL** with our database for general fullerenes, **IPR** with our database for IPR fullerenes, **Yoshida** with Yoshida's database of selected fullerenes (directories containing .cc1 files), and **HOG** with (if downloaded from <http://hog.grinvin.org/Fullerenes>) the House of Graph data files. Important: Please make all filenames in the database read-only. The files downloaded from our website are gzipped, so you need to gunzip the file you like to use before executing the program.

To use the latex program for the 2D fullerene graphs you need to use the files **standalone.cfg** and **standalone.cls**.

3 Program Structure

The main program (`main.f`) calls a number of subroutines for certain tasks and in a certain sequence (see Figure 2). Subroutine **DataIn** manages all the input and determines these main tasks to be carried out. The input is in *Namelist format* if not otherwise stated. Important steps in the program are given in Figure 2 and are explained in some detail in this section. Figure 3 visualizes the workflow in a more abstract way. More information can be obtained in our review article [11].

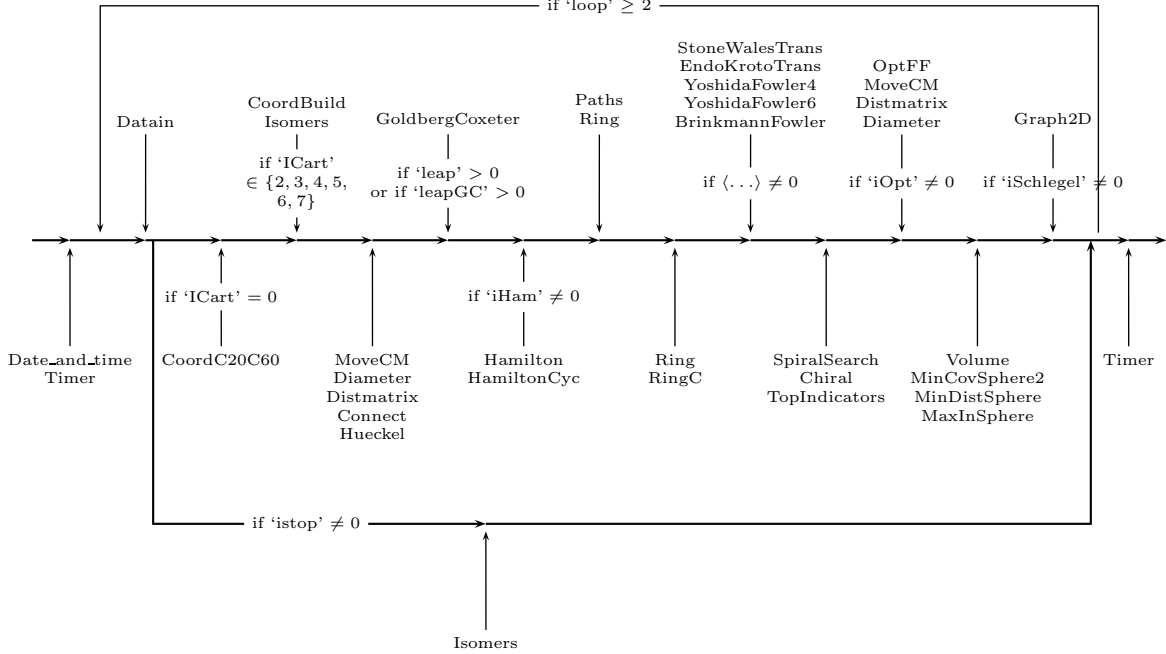


Figure 2: Flow diagram for main program tasks.

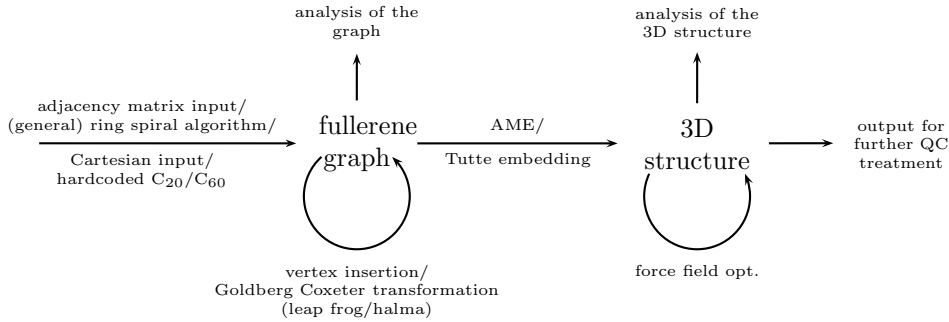


Figure 3: Simplified diagram of the workflow.

The most time-consuming steps in the program are the diagonalization of the adjacency matrix of the fullerene graph F , $A(F)$ (needed for the AME algorithm and the Hückel analysis ($\mathcal{O}(n^3)$)), the force-field optimization for very large fullerenes (> 2000 atoms) and the diagonalization of the force-field Hessian, the determination of the number of Hamiltonian cycles (> 80 atoms) (this is a $\#P$ -hard problem), the creation of a list of ring-spiral pentagon indices for all possible isomers with a given vertex count n_v (> 80 atoms), and the Tutte embedding algorithm (> 2000 atoms). We try to improve the performance of these algorithms in our next major version to be released. For version 4.1 the fullerene database was increased and a link to the House of Graphs was introduced. Version 4.2 saw the introduction of dihedral angles into the force field, and Version 4.3 was extended to calculate frequencies from the force-field Hessian, and Version 4.4 saw the implementation of the Goldberg-Coxeter transformation $GC_{k,l}$ of C_{20} for all possible combinations of $k \geq l$, new topological descriptors and a more compact version of the database. Version 4.5 removed some of the bugs and includes a general Goldberg-Coxeter transformation $GC_{k,l}[C_n]$ ($k \geq l, l \geq 0$) of any fullerene. Version 4.5 also has optimized force constants for a frequency analysis.

3.1 Create a 3D structure for a specific fullerene

The easiest way to create a 3D structure for *Fullerene* is of course to read-in cartesian coordinates for a specific fullerene (see input files `CCc20.inp` to `CCc540.inp`), or to construct them internally for the high symmetry I_h-C_{20} or I_h-C_{60} isomers (Subroutine **CoordC20C60**, see input files `GEOMc20.inp`, `GEOMc60.inp`, `GEOMc60exp.inp`, `GEOMc60ideal.inp`), or to construct the cartesian coordinates from a generalized ring-spiral pentagon-indices RSPI (plus jumps if necessary) by using either the Fowler-Manolopoulos AME algorithm [9], or the more reliable 3D-TE algorithm (Subroutine **CoordBuild**, see input files `RSPIc60Ih` or `RSPIc840-D5.inp`), or from a Goldberg-Coxeter 2-index transformation of for example C_{20} , i.e. $GC_{k,l}(C_{20})$ with $k \geq l$ and $k > 0$ (triangulation algorithm described below, see input files `GCK2L1c20.inp`, `GCK4L3C20.inp` or `GCK5L3c20.inp`). The 12 pentagon indices can also be obtained from the *fullerene database* using the canonical fullerene number as defined in the book “Atlas of Fullerenes” by Fowler and Manolopoulos [9, 37] (e.g. input file `DBc66.inp`). Information of vertex connections (edges) can also be used as input (input files `EDGEc20.inp`, `EDGEc60.inp` and `EDGEc440.inp`). The program prints internal coordinates in form of a Z-Matrix (distances, angles and dihedrals). For this the atoms are rearranged (permuted) such that the distances are smallest. Note that this works well for optimized coordinates, e.g. cartesian coordinates obtained from a force-field optimization.

In addition to the original ring spiral algorithm by Fowler and Manolopoulos we have implemented a more general ring spiral algorithm for cubic graphs. It differs from the classical ring spiral algorithm in that it can insert jumps between the addition of two vertices to the spiral in order to prevent walking into a cul-de-sac at a later stage. It is therefore capable of describing all fullerene graphs (and any other cubic graph). The criterion for jumping (while deriving the spiral from a graph) is, that the subgraph that hasn’t been added to the spiral yet, must never be disconnected. For deriving the graph from the spiral it is necessary to explicitly give the jump positions and distances.

We recommend reading Fowler and Manolopoulos’ book [9, 37] for the use of RSPIs and Hückel (adjacency matrix) P -type eigenvectors to construct 3D fullerene graphs. For the Goldberg-Coxeter transformation of C_{20} we obtain the RSPI from the scheme introduced by Fowler and Rogers.[39] For the AME algorithm it is critical to get the right Hückel eigenvectors for the construction of cartesian coordinates,[9] which works well for the icosahedral fullerenes, but less well for fullerene nanotubes. The position of the three P -type eigenvectors can be specified. If the AME algorithm fails or you cannot find the required P -type eigenvectors, you should use our Tutte embedding algorithm, which (to our opinion) is less troublesome and may be used as the standard method to construct 3D structures. The disadvantage of this algorithm is, however, that the structure could be far off from a force-field optimized global minimum for a fullerene, and the force-field optimization might end up in a heavily distorted unreasonable 3D local minimum (for strategies to avoid this see the input section).

It is important that the end-product is viewed by a molecular visualization program. We recommend CYLview by Claude Legault [30], Avogadro [31], Jmol [32], Pymol [33] or VMD [1]. Most of these programs are freely available. For this purpose a file is written to `Fullerene-3D.xyz` or `Fullerene-3D.cc1` to be used as an input file for the various molecular visualization programs available.

The program sets the barycenter of the fullerene to the origin. Using the obtained cartesian coordinates the program calculates the smallest and largest cage diameters and the moment of inertia, which already gives a measure for distortion from spherical symmetry (Subroutine **Diameter**). It produces the distance matrix d_{ij} if the print level is set to high (Subroutine **DistMatrix**), and once the molecular point group is obtained, the program determines if the fullerene is chiral or not (Subroutine **Chiral**). Once the cartesian coordinates (X_i, Y_i, Z_i) for each vertex $i = 1, \dots, n_v$ is known, all edges are determined by analyzing connectivities between the vertices from either given cartesian coordinates or directly from the adjacency matrix if a ring spiral input was chosen (Subroutine **Connect**). In any case, one has the adjacency matrix A_{ij} for all vertices at this stage:

$$A_{ij} = \begin{cases} 1 & \text{if vertices } i \text{ and } j \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

As the adjacency matrix is symmetric, the eigenvalues are all real.

3.2 List of Isomers

The program can print all general and IPR isomers for a specific fullerene and perform an analysis as introduced in the book by Fowler and Manolopoulos [9], e.g. pentagon indices and pentagon numbers, hexagon indices and strain parameter, NMR information and number of distinct Hamiltonian cycles if required. The number of isomers scales polynomially as n_v^9 , thus it becomes computationally very demanding to produce a full isomer list for fullerenes larger than C_{120} . Moreover, the database provided, can be used to print all isomers up to C_{150} , and up to C_{200} for all IPR isomers. Fowler and Manolopoulos use a numbering scheme ($N_{\text{isomers}} = 1, \dots, n_{\text{max}}$) obtained from the natural sequence of canonical RSPs to distinguish between the different isomers [9]. From their book [9] the general isomer number can be taken and the RSPs for a specific isomer up to C_{150} can be read-in from the database provided. The numbering scheme for IPR isomers only ($N_{\text{isomers}}^{\text{IPR}} = 1, \dots, n_{\text{max}}$) can also be used up to C_{200} , see for example the input file `DBc50.inp`.

3.3 Hückel analysis and topological indicators

From the adjacency matrix A_{ij} of eq.(1), a Hückel analysis is performed (Subroutine `Hueckel`). This gives you a good hint if the fullerene is of open or of closed shell nature [9]. For smaller fullerenes or non-IPR fullerenes, the Hückel analysis may not be very reliable as hybridization with the $C(2s)$ orbitals occurs due to non-planarity. Hence the $\sigma - \pi$ separation breaks down. The orbital energies are defined as

$$\epsilon_i = \alpha + x_i \beta \quad (2)$$

and we adopt $\alpha = -0.21$ au and $\beta = -0.111$ au obtained from the experimental ionization potential (7.58 eV) and excitation energy (3.02 eV) of C_{60} (NB: the electron goes into the second LUMO of t_{1g} symmetry). This gives also orbital energies for C_{60} in reasonable agreement with DFT Kohn-Sham orbital energies. The fullerene structure might also undergo a Jahn-Teller distortion adopting a singlet ground state instead of one of a higher multiplicity, as this is the case for C_{20} . Such electronic effects are not captured in this program, and one needs to perform a proper quantum-theoretical calculation. The program gives information on the HOMO-LUMO gap and calculates the topological resonance energy from Babić's matching polynomial calculations [24, 25].

A topological index is a structurally invariant natural or real number related to a molecular graph, which does not depend on the labelling or the 2D representation of a graph G . In a more abstract way, a topological index is a map $\rho : G \rightarrow \{a_i\}$ from a graph G into a set of numbers a_i (integers, rational or real numbers). There are several well known topological indicators [35, 36] of which the Wiener index, the hyper-Wiener index, Estrada index, the Balaban index, the Szeged index, the topological radius, diameter and mean distance are printed.

The Estrada $E(G)$ and bipartivity $\beta(G)$ indices are obtained from the eigenvalues of the adjacency matrix $A(G)$, $\{x_i, i = 1, \dots, n_v\} \in [-3, 3]$, of a graph G (in our case a regular fullerene graph $G = F$) [40, 41],

$$E(G) = \sum_{i=1}^{n_v} e^{x_i} \quad \text{and} \quad \beta(G) = \sum_{i=1}^{n_v} \cosh(x_i) / E(G) \quad (3)$$

The Estrada index for I_h - C_{60} is $E = 197.450228$ and the corresponding bipartite index $\beta = 0.992966$.

The Wiener $W(G)$ and hyper-Wiener $WW(G)$ indices are defined as [27],

$$W(G) = \frac{1}{2} \sum_{\{i,j\} \subseteq V(G)} D_{ij} \quad \text{and} \quad WW(G) = \frac{1}{2} \sum_{\{i,j\} \subseteq V(G)} (D_{ij} + D_{ij}^2) \quad (4)$$

Here, $V(G)$ is the vertex set and D_{ij} is the topological (chemical) distance matrix containing the topological distances between the two vertices i and j , i.e., the number of edges in the shortest path connecting them.

For I_h - C_{60} we have $W(G) = 8340$ and $WW(G) = 27180$. If we define

$$d_i^{\max} = \max_j \{D_{ij}\}, \quad (5)$$

we can define the topological diameters D and radius R as [36]

$$D = \max\{d_i^{\max}\} \quad \text{and} \quad R = \min\{d_i^{\max}\}. \quad (6)$$

We can now define the reverse Winer index[42] as

$$W_{\text{rev}} = n_v(n_v - 1)D/2 - W \quad (7)$$

which for C_{60} is 7590. The Balaban index is defined as [28],

$$B(G) = \frac{n_e}{n_e - n_v + 2} \sum_{i < j \in E(G)} (W(i)W(j))^{-1/2} \quad (8)$$

n_e are the number of edges and $W(i)$ is the sum of topological distances between vertex i and all other vertices in the graph,

$$W_i = \sum_{j \in V(G)} D_{ij} \quad (9)$$

For $I_h\text{-}C_{60}$ we have $B(G) = 0.910521583$. The Schultz and Zagreb indices are also listed, although they are related to the indices already defined here. The Szeged index is defined as [43]

$$S_z(G) = \sum_{e \in E(G)} n_{i_j}(e)n_{j_i}(e) \quad (10)$$

where $n_{i_j}(e) = |B_{i_j}(e)|$ is the number of vertices which are closer to vertex i than vertex j in an edge $e = (ij) \in E(G)$ (with $E(G)$ being the edge set) and

$$B_{i_j}(e) = \{k | k \in V(G), D_{ik} < D_{jk}\} \quad (11)$$

Further listed are two topological efficiency parameters ρ and ρ_E defined by Vukicevic, Ori and co-workers [44, 46],

$$\rho = 2 \frac{W(G)}{n_v W_{\min}} \quad \text{with} \quad W_{\min} = \min\{W_i\} \quad (12)$$

and

$$\rho_E = \frac{W_{\max}}{W_{\min}} \quad \text{with} \quad W_{\max} = \max\{W_i\} \quad (13)$$

For $I_h\text{-}C_{60}$ both ρ and ρ_E are exactly 1.0. The spectral moments $\{\mu_k, k = 0, 1, \dots\}$ are calculated from eigenvalues x_i of the adjacency matrix,

$$\mu_k = \sum_{i=1}^{n_v} x_i^k \quad (14)$$

and we have $\mu_0 = n_v$, $\mu_1 = 0$, $\mu_2 = 3n_v$, $\mu_3 = 0$, $\mu_4 = 15n_v$, $\mu_5 = 120$, $\mu_6 = 93n_v - 120$, and $\mu_7 = 1680$. The higher spectral moments depend on the fullerene structure.[45]

One of the first topological indicators introduced to discuss the stability of fullerenes are the neighbour indices for pentagons and hexagons. The program calculates the pentagon neighbour indices,[9, 48] hexagon neighbour indices and the pentagon arm indices.[47] Every fullerene isomer can be characterized by a signature of the form $\{i_k | k = 1, \dots, n\}$. The pentagon indices $(p_i | i = 0, \dots, 5)$ define the number of pentagons attached to another pentagon, i.e., for IPR fullerenes $p_0 = 12$ and all other pentagon indices are zero. Hexagon indices are similarly defined, i.e., $(h_i | i = 0, \dots, 5)$, where h_k is the number of hexagons with neighbour index k . In an IPR fullerene every hexagon is adjacent to a minimum of three others and we can restrict the list to (h_3, h_4, h_5, h_6) . The pentagon arm indices $(n_i | i = 0, \dots, 5)$ counts the number of arms of the pentagons. Here an edge E incident to a vertex of a pentagon not belonging to the pentagon is called an arm if i) both end-vertices of edge E are incident to pentagons, and ii) E shares two neighbour hexagons.[47] It is clear that this is nothing else than a Stone-Wales pattern.

We can now define some useful topological invariants. The pentagon index N_p is defined as

$$N_p = \frac{1}{2} \sum_{k=1}^5 k p_k \quad \text{with} \quad \sum_{k=0}^5 p_k = 12 \quad (15)$$

For IPR fullerenes we have no pentagon attached to another and therefore $p_0 = 12$ and all other $p_i = 0$, and it follows that $N_p = 0$. For IPR fullerenes a more useful index is defined through the hexagon neighbour indices. The standard deviation σ_h of the hexagon neighbour index distribution is defined as

$$\sigma_h = \sqrt{\langle k^2 \rangle - \langle k \rangle^2} \quad (16)$$

where

$$\langle k^n \rangle = \frac{\sum_{k=1}^6 k^n h_k}{\sum_{k=0}^6 h_k} \quad \text{with} \quad \sum_{k=0}^6 h_k = \frac{n}{2} - 10 \quad (17)$$

Réti and László define the pentagon arm index N_A in a similar way to the pentagon index,[47]

$$N_A = \frac{1}{2} \sum_{k=1}^5 k n_k \quad \text{with} \quad \sum_{k=0}^5 n_k = 12 \quad (18)$$

The first and second moments M_1 and M_2 and the variance Var are defined as (real numbers),

$$M_i = \frac{1}{12} \sum_{k=1}^5 k^i n_k \quad \text{and} \quad \text{Var} = M_2 - M_1^2 \quad (19)$$

We now define the Réti-László topological descriptor as Ψ ,

$$\Psi = \frac{30 + 6M_1}{1 + 4.5N_p + C(M_1, M_2)} \quad \text{with} \quad C(M_1, M_2) = \frac{(120M_2)^{1/2}}{(1 + 7M_1)^{1/2}(1 + 0.9\text{Var}^{1.5})} \quad (20)$$

If there exists a non-negative integer q for which one of the arm indices $n_q = 12$, the fullerene is called q -balanced. In this case $\text{Var} = 0$ [47]. If $N_p + N_A = 0$, the fullerene is called strongly isolated. C_{60} contains Stone-Wales patterns and is therefore not strongly isolated. The first leapfrog of C_{60} , C_{180} , and the ones to follow are strongly isolated.

The program also calculates the number of perfect matchings in polynomial time using the algorithm of Fisher, Kasteleyn, and Temperley [38]. The algorithm uses a **Pfaffian** computation of a skew-symmetric matrix derived from a planar embedding of the graph. To use this you need **lapack** installed on your computer and change the **Makefile**.

3.4 The Goldberg-Coxeter transformation

In a next step a Goldberg-Coxeter transformation $GC_{k,l}(n)$ for a fullerene C_n can be performed if required from the input (Subroutine **GoldbergCoxeter**). This leads to a fullerene $C_{n(k^2+kl+l^2)}$. The current implementation allows not only for *leapfrog transformations* ($k = l = 1$) and *halma transformations* ($k \neq 0$ and $l = 0$) for any fullerenes as implemented in version 4.4, but also now for all indices and fullerenes, i.e. $GC_{k,l}(C_n)$ with $k \geq l$.

The general Goldberg-Coxeter construction $GC_{k,l}[G_0]$ used here and shown in Figure 4 works in the following way: (i) Consider a cubic planar graph G_0 and take its dual (faces become vertices and vertices become faces), $D_t(G_0)$. This transforms the graph G_0 with n_v vertices into a triangulation, i.e. a geodesic sphere with n_v triangles; (ii) Each triangle T_k ($k = 1, \dots, n_v$) of the geodesic sphere is tiled into another set of faces t_i using the Coxeter indices (k, l) . If the obtained new smaller faces t_i are not triangles themselves within one T_k , when neighbouring T_k s are glued together with other non-triangle faces, new triangles are formed and we end up with a new triangulation, i.e. a larger geodesic sphere $GC_{k,l}D_t(G_0)$; (iii) Finally we take the dual which yields a new fullerene,

$$G_1 = GC_{k,l}[G_0] = D_t[GC_{k,l}D_t(G_0)] \quad (21)$$

If the initial graph G_0 has n_v vertices, the number of vertices of $GC_{k,l}[G_0]$ is

$$n_v^{GC} = t(k, l)n_v = (k^2 + kl + l^2)n_v \quad (22)$$

$t(k, l)$ is called the *triangulation number*. We work with the dual triangulation instead of a hexagonal mesh as it is easier to be implemented in a computer code. This is shown in Figure 4. For icosahedral fullerenes (I_h and I) with $l = k$ or $l = 0$, Fowler and Rogers showed how to construct the RSPIs,[39] which is implemented in this program as well for $GC_{k,l}(C_{20})$. Figure 5 shows a $GC_{2,1}[C_{80}]$ transformation of a D_{5d} - C_{80} carbon nanotube transforming it into a chiral nanotube.

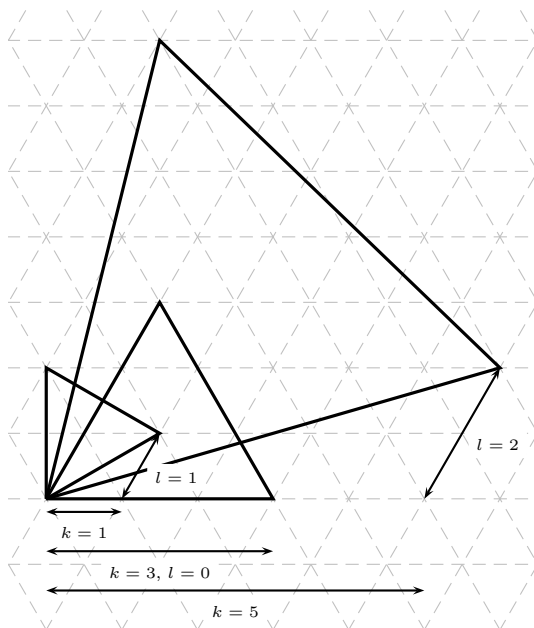


Figure 4: Goldberg-Coxeter construction: $GC_{1,1}$, leapfrog transformation; $GC_{3,0}$, halma transformation; $GC_{5,2}$, general.

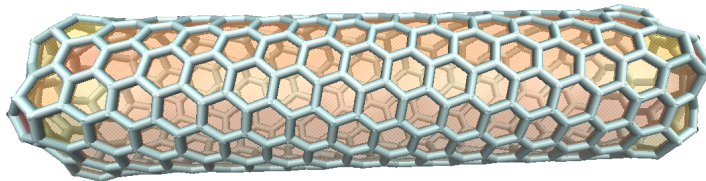


Figure 5: Goldberg-Coxeter transformation $GC_{2,1}[C_{80}]$ transformation of a D_{5d} - C_{80} carbon nanotube into the chiral D_5 - C_{560} nanotube.

3.5 Hamiltonian cycles

Subroutine **Hamilton** uses the back-track algorithm of Babić [49] to obtain all Hamiltonian cycles and subsequently (if required) the IUPAC name of the fullerene. Hamiltonian cycles go through all vertices exactly once returning to the starting point. The number of distinct (non-isomorphic) Hamiltonian cycles has carefully been checked against a second algorithm developed by us for various fullerenes. Left-right cycles and cyclic vertex permutations count as the same cycle as they are isomorphic to each other. Although finding all Hamiltonian cycles is an NP-complete problem, the algorithm works fine up to about C_{100} . After that it becomes computationally very demanding as the algorithm scales like $O(2^{n_v})$. Therefore, for the larger fullerenes the program prints tight upper and lower limits for the number of Hamiltonian cycles instead. The existence of Hamiltonian cycles for fullerenes is only conjectured at this stage, and only for layered fullerenes (e.g. fullerene with onion-shaped 2D graphs such as nanotubes) existence has been proven. Our recent calculations verified that Hamiltonian cycles exist for all fullerene isomers up to C_{120} and for all IPR isomers up to C_{122} [50].

3.6 Force-field optimization

The fullerene structure can be optimized by force fields using a Fletcher-Reeves-Polak-Ribiere geometry optimization [19] with analytical gradients (Subroutine **OptFF** [51]). In the current version the harmonic oscillator force field (HOFF) is implemented, which considers both bond lengths and angles in an harmonic

oscillator approximation [20]:

$$E_{\text{Wu}} = \frac{k_p}{2} \sum_{i_p}^{\text{p-edges}} (R_{i_p} - R_p)^2 + \frac{k_h}{2} \sum_{i_h}^{\text{h-edges}} (R_{i_h} - R_h)^2 + \frac{f_p}{2} \sum_{j_p}^{60} (\theta_{j_p} - \theta_p)^2 + \frac{f_h}{2} \sum_{j_h}^{3 \times n_v - 60} (\theta_{j_h} - \theta_h)^2 \quad (23)$$

Only bonded pairs of vertices are taken into account. k_p and k_h are the force constants for the two different C-C bonds (set to $\approx 300 \text{ kcal } \text{\AA}^{-2}$), R_p and R_h the corresponding pentagon and hexagon bond distances ($\approx 1.4 \text{ \AA}$), and θ_p and θ_h are the corresponding bond angles (108° and 120° respectively). As the original Wu force field was developed for $I_h\text{-C}_{60}$ only, adjacent pentagons are treated in the same way as a pentagon adjacent to a hexagon.

The HOFF optimization is very fast even for fullerenes such as C_{840} . The HOFF force-field optimization may lead to distortions of the fullerene to a structure of lower symmetry compared to the original point-group. Moreover, as no dihedral angles are fixed, the molecule might distort away from convexity or vertices are not placed correctly leading to edge crossings. In such cases an additional Coulomb repulsive potential can be added (see input instructions) in a preoptimization step to avoid large “dents” or edge crossings of the fullerene,

$$E = E_{\text{Wu}} + E_{\text{Coulomb}} = E_{\text{Wu}} + \sum_{i=1}^{n_v} \frac{f_{\text{Coulomb}}}{|\vec{r}_i - \vec{r}_0|} \quad (24)$$

with $\vec{r}_i - \vec{r}_0$ being the distance between vertex i and the barycenter at \vec{r}_0 . Another method to influence convergence into the global minimum is to adjust the sphere radius in the Tutte embedding algorithm (see input section). Note that the construction of the fullerene by using the AME or Tutte algorithm may lead to a more spherical arrangement prior to a force-field optimization with rather large bond distances, e.g. barrels instead of nanotubes.

A more sophisticated force field using dihedral angles is now also implemented (extended HOFF force field), which enhances planarity for areas of connected hexagons.[21] The extended Wu force field takes three types of bonds (adjacent to 0, 1 or 2 pentagons), two types of angles, and four types of dihedral angles into account. There is one dihedral per atom. Dihedrals θ_{abcd} are defined between one atom a and its three neighbours b , c and d . As one atom is part of three faces (0, 1, 2, or 3 pentagons) there are four different types of dihedrals which differ in their respective zero value and force constant.

The total energy for the extended Wu force field is given by

$$E_{\text{extWu}} = \frac{f_{pp}}{2} \sum_{i_{pp}}^{\text{pp-e}} (R_{i_{pp}} - R_{pp})^2 + \frac{f_{hp}}{2} \sum_{i_{hp}}^{\text{hp-e}} (R_{i_{hp}} - R_{hp})^2 + \frac{f_{hh}}{2} \sum_{i_{hh}}^{\text{hh-e}} (R_{i_{hh}} - R_{hh})^2 + \frac{f_p}{2} \sum_{j_p}^{60} (\theta_{j_p} - \theta_p)^2 + \frac{f_h}{2} \sum_{j_h}^{3n_v-60} (\theta_{j_h} - \theta_h)^2 + \frac{f_{ppp}}{2} \sum_{k_{ppp}}^{\text{ppp-v}} (\theta_{k_{ppp}} - \theta_{ppp})^2 + \frac{f_{hpp}}{2} \sum_{k_{hpp}}^{\text{hpp-v}} (\theta_{k_{hpp}} - \theta_{hpp})^2 + \frac{f_{hhp}}{2} \sum_{k_{hhp}}^{\text{hhp-v}} (\theta_{k_{hhp}} - \theta_{hhp})^2 + \frac{f_{hhh}}{2} \sum_{k_{hhh}}^{\text{hhh-v}} (\theta_{k_{hhh}} - \theta_{hhh})^2 \quad (25)$$

where pp-e (hp-e, hh-e) denotes the number of edges adjacent to 2 (1, 0) pentagons, n_v is the number of vertices and ppp-v (hpp-v, hhp-v, hhh-v) is the number of vertices between 3 (2, 1, 0) pentagons. The additional Coulomb force, that can be added to the extended HOFF field, is equal to the aforementioned optional Coulomb force defined in eq.(24). The force constants have been adjusted to vibrational frequencies of several fullerenes obtained from density functional calculations, which, for example, leads to reasonably good zero-point vibrational energies.[21]

3.7 Ring connectivities, patterns, vertex insertions and spiral detection

Subroutine **Ring** identifies all pentagons and hexagons (faces) and checks if Euler’s theorem is fulfilled. Subroutine **RingC** then determines the center for each pentagon and hexagon used later for the trigonal pyramidal tessellation to obtain the fullerene volume and surface. This routine also analyzes all 2- and

3-ring fusions and many other useful patterns needed for example in Stone-Wales transformations or vertex insertions. It further determines the Rhagavachari-Fowler-Manolopoulos neighbouring pentagon and hexagon indices as described in detail in Fowler and Manolopoulos' book [9]. From the hexagon indices one derives if the fullerene is IPR or not. If it is IPR, Cioslowski's scheme is used to calculate the heat of formation [23]. For all fullerenes, Martin's scheme is used to calculate the heat of formation [26].

Information from subroutine **RingC** can be used to perform Stone-Wales transformations [18] (subroutine **StoneWalesTrans**), Endo-Kroto 2-vertex insertions [15] (subroutine **EndoKrotoTrans**), Yoshida-Fowler 4-vertex (D_{3h} 6555) (subroutine **YoshidaFowler4**) and 6-vertex (D_{3h} 666555) insertions (subroutine **YoshidaFowler6**) [16], or a Brinkmann-Fowler 6-vertex (6-55-55) insertion (subroutine **BrinkmannFowler**) [17], see for example input files `CCc50EK.inp`, `GEOMc60SW.inp`, `CCc60YF.inp`. The vertex-insertion nomenclature introduced by Brinkmann and Fowler [17] defines these insertions uniquely:

Endo-Kroto: 2 pentagon G2.12.1.1 ($n_v=12$) \rightarrow G2.12.1.2 ($n_v=14$);

Yoshida-Fowler 4 vertex insertion: 3 pentagon G3.15.3.1 ($n_v=15$) \rightarrow G3.17.3.2 ($n_v=19$);

Yoshida-Fowler 6 vertex insertion: 3 pentagon G3.15.4.1 ($n_v=19$) \rightarrow G3.15.4.2 ($n_v=21$) followed by G3.15.3.1 ($n_v=15$) \rightarrow G3.15.3.2 ($n_v=19$);

Brinkmann-Fowler 6-vertex insertion: 4 pentagon G4.14.2.1 ($n_v=16$) \rightarrow G4.14.2.2 ($n_v=22$).

Non-spiral fullerenes such as C_{380} or C_{384} can be constructed with such vertex insertion methods, see input files `RSPIC380.inp` and `RSPIC384.inp`. The two structures obtained from a force-field optimization are shown in Figure 6 using program PYMOL [33].

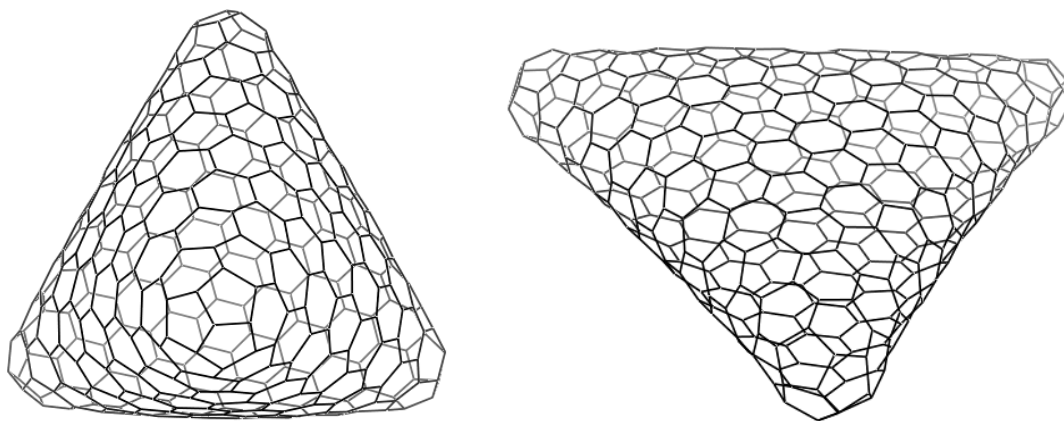


Figure 6: Optimized Wu force-field structures of C_{380} (left) and C_{384} (right), the first fullerenes with no ring spiral.

Subroutine **SpiralSearch** uses the ring-spiral algorithm of Fowler and Manolopoulos [9] for the search of a ring spiral if, for the input, cartesian coordinates or vertex connections (adjacencies) were chosen, or if the initial fullerene structure was modified. It produces the canonical ring-spiral pentagon indices if the fullerene contains a face spiral, and fails if it does not (as for the two examples shown in Figure 6). Note that considerable improvements were made to optimize the ring spiral search algorithm, and if **ispsearch**=2 is chosen it counts all possible ring spirals (this is computationally more expensive). The number of spirals in the output agree with the examples given by Yoshida and Fowler.[16] A generalized ring-spiral algorithm has been implemented, and if the search for a simple ring spiral algorithm fails, RSPIs are produced with additional jumps using our general spiral algorithm.[53]

3.8 Volume V and surface area A , minimum covering sphere, minimum distance sphere and maximum inner sphere

The volume and surface area of the fullerene are calculated in Subroutine **Volume**. This is done by

- 1) The convex hull.
- 2) The normal vector algorithm resulting in the exact volume of the polyhedron (identical to convex hull if polyhedron is convex).
- 3) The tessellation into trigonal pyramids from the barycenter by summing over all tetrahedrons spanned by the three vectors (identical to algorithm 1 and 2 if the polyhedron is convex). A sum over all tetrahedrons spanned by the three vectors CM-CR (center(mass)-center(ring): barycenter of fullerene to the ring center), CM-CA1 (barycenter of fullerene to atom 1 in ring), and CM-CA2 (barycenter of fullerene to atom 2 in

ring) is calculated. There are 5 such tetrahedrons in a pentagon and 6 in a hexagon. The barycenter CM defines the origin of the 3D fullerene structure. In a similar way the surface area A is obtained by summing over all triangles from the ring center to neighbouring vertices in the ring. We note that the I_h -C₂₀ and I_h -C₆₀ coordinates can be constructed easily using basic geometry. For these, the volume and surface areas are known analytically.

The *isoperimetric quotient* (IPQ) is defined as

$$q_{\text{IPQ}} = 36\pi \frac{V^2}{A^3} \quad \text{with} \quad q_{\text{IPQ}} \in [0, 1] \quad (26)$$

which for an ideal sphere is trivially $q_{\text{IPQ}} = 1$, and for zero volume $q_{\text{IPQ}} = 0$. For C₂₀ and C₆₀ with equal bond lengths we get

$$q_{\text{IPQ}}(\text{C}_{20}) = 0.75470 \quad (27)$$

$$q_{\text{IPQ}}(\text{C}_{60}) = 0.90317 \quad (28)$$

We can now define the deviation from a sphere (in %) by

$$D_{\text{IPQ}} = 100(1 - q_{\text{IPQ}}) \quad (29)$$

The program also calculates the *sphericity parameter* of Diaz-Tendero [54],

$$q_{\text{SP}} = \sqrt{(a-b)^2 + (a-c)^2 + (b-c)^2} / A \quad (30)$$

where $a \geq b \geq c$ are the rotational constants. Note that we deviate from the original definition by dividing the square-root by the rotational constant A . Further, the *asymmetry parameter* of Fowler is printed,

$$q_{\text{AP}} = \sum_{i=1}^{n_v} \frac{(r_i - r_{av})^2}{r_{av}^2} \quad (31)$$

where r_i is the radial distance of atom i from the barycenter and r_{av} is the average distance [55].

Three routines follow to calculate the *minimum covering sphere* (MCS) of a fullerene (Subroutine **MinCovSphere2**) using algorithm 2 of Yildirim [56], *minimum distance sphere* (MDS) (Subroutine **MinDistSphere**), and the *maximum inner sphere* (MIS) (Subroutine **MaxInSphere**). The MCS is defined as follows: Let $S = \{\vec{p}_i\}$ be the set of n points ($i = 1, \dots, n$) in m -dimensional Euclidean space, \mathbb{R}^m . The MCS of this set, $\text{MCS}(S)$, is a sphere of smallest radius R_{MCS} that encloses the set of points S , and can be expressed as follows,

$$\min_{\vec{c}_{\text{MCS}}} \max_i \|\vec{p}_i - \vec{c}_{\text{MCS}}\| \quad (32)$$

where $\|\cdot\|$ denotes the Euclidean norm and \vec{c}_{MCS} is the center of the MCS. The MCS is uniquely defined and can be expressed as a convex combination of at most $(m+1)$ points, hence our algorithm stops when 4 points (3D space with $m = 3$) are left over in the iteration process of the algorithm. The spherical central cover SCC is usually not the minimum covering sphere MCS (except for example if all distances from the barycenter CM are the same as in the ideal capped icosahedron). The spherical central cover is taken from the CM point with radius R_{max} (longest distance to one vertex). We changed the first condition in Yildirim's algorithm by choosing R_{max} as the furthest point from CM. In the final statistics there should be 0 points outside the sphere and at least 1 point on the sphere. We can now introduce the definition for the MCS distortion parameter D_{MCS} (in % of r_{min}),

$$D_{\text{MCS}} = \frac{100}{n_v r_{\text{min}}} \sum_{i=1}^{n_v} (R_{\text{MCS}} - \|\vec{p}_i - \vec{c}_{\text{MCS}}\|) \quad (33)$$

At the end, the Van der Waals radius of carbon (1.415 Å) is added to the radius of the MCS (input coordinates for this need to be in Å otherwise you are required to change the program), and the volume of an ideal fcc solid is calculated. The Van der Waals radius is chosen such that for C₆₀ the solid-state results of Heiney et al. [58] are reproduced. In the case of nonplanar pentagons or hexagons there is no unique definition for the volume of a fullerene, except for the convex hull. There is no reason, however, why any other definition should be preferred over the exact volume algorithm employed in this program.

The MCS is biased for the case that few atoms stick out on the surface of a fullerene, and the minimum distance sphere (MDS) may be more appropriate for a measure from spherical distortion,

$$\min_{c_{\text{MDS}} \in \text{CH}(S)} \frac{1}{n_v} \sum_i |R_{\text{MDS}} - \|\vec{p}_i - \vec{c}_{\text{MDS}}\|| \quad (34)$$

with the MDS radius

$$R_{\text{MDS}} = \frac{1}{n_v} \sum_i \|\vec{p}_i - \vec{c}_{\text{MDS}}\| \quad (35)$$

The restriction of \vec{c}_{MDS} to be inside the convex hull is necessary as \vec{c}_{MDS} may move out of the fullerene structure in the optimization procedure. In general we have $\vec{c}_{\text{MDS}} \neq \vec{c}_{\text{MCP}}$ except for the case that all points lie on the MCS. The MDS may however not be uniquely defined, as there could be many (even degenerate) local minima, but for most cases (i.e. "spherical" fullerenes) it works just fine. This gives a better definition for the distortion parameter compared to D_{MCS} , as it is not biased to a few points lying outside or inside an ideal fullerene sphere. Hence, analogous to the MCS we define a measure for distortion from spherical symmetry through the MDS,

$$D_{\text{MDS}} = \frac{100}{n_v r_{\text{min}}} \sum_{i=1}^N |R_{\text{MDS}} - \|\vec{p}_i - \vec{c}_{\text{MDS}}\|| \quad (36)$$

The maximum inner sphere (MIS) is important to estimate if there is enough space inside the fullerene cage for encapsulating endohedral atoms or molecules. It is defined as

$$R_{\text{MIS}} = \max_{c_{\text{MIS}} \in \text{CH}(S)} \min_i \|\vec{p}_i - \vec{c}_{\text{MIS}}\| \quad (37)$$

For ideal C_{60} (I_h symmetry) MCS, MDS and MIS are all identical. Note that the MIS may not be uniquely defined. For example, for nanotubes there may be degenerate solution with different \vec{c}_{MDS} along the main axis of the nanotube. For the MIS the radius and volume is printed with the Van der Waals radius of carbon taken off R_{MIS} .

3.9 2D fullerene embeddings and Schlegel diagrams

Fullerenes can be nicely visualized by 2D representations. Such embeddings should avoid edge crossings as fullerene graphs are planar, and should represent the point group symmetry as best as possible. There are a number of different algorithms available trying to achieve this, and in subroutine **Graph2D** (X_i, Y_i) ($i = 1, \dots, n_v$) coordinates for a 2D fullerene graph are created. Here we briefly describe the algorithms we recommend to use, others are of perhaps of more historical interest. Note that for non-spherical fullerenes (like the famous non-spirable C_{380} (Figure 6) tetrahedron) it becomes very difficult to create a good Schlegel projection, and in such cases the projection of vertices to the minimum covering sphere is recommended before these algorithms are used. The subroutine produces a latex file with the fullerene graph included.

1) In the perspective Schlegel projection (PSP), the 3D graph is rotated such that a selected vertex, edge of ring (commonly the barycenter of a polygon is chosen) is located at top of the z -axis at some distance below the projection point P , with the z -axis going (not necessarily) through the barycenter of the fullerene. If no choice is given in the input, the point $(0, 0, z_{\text{max}})$ is chosen with z_{max} being the point with maximum z -value from the original input coordinates of the vertices. Vertices and ring barycenters are then sorted in descending order according to their z -values. The Schlegel projection then yields the projected (X_i, Y_i) coordinates by projecting vertices and ring centers down on a plane below the fullerene. The edges between the vertices are already known such that the fullerene graph can be drawn. This is shown in Figure 7.

2) In the cone "Schlegel" projection (CSP) the vertices are projected out on an enveloping cone and then down on a plane below the fullerene (see Figure 7 (b)). The barycenter of the last ring closest to the projection plane should be at the bottom of the fullerene. The barycenter of the circumfencing ring will not be projected out (this center may be ignored in the drawing). A scaling factor could be applied (1.2 used in the program) for the outer points in the 2D graph in order to avoid edge crossings.

At the end a rough plot of the fullerene graph is produced. This is suitable for fullerenes up to about C_{100} , beyond that it becomes too crowded and the latex file produced contains a better graph. Nevertheless, it

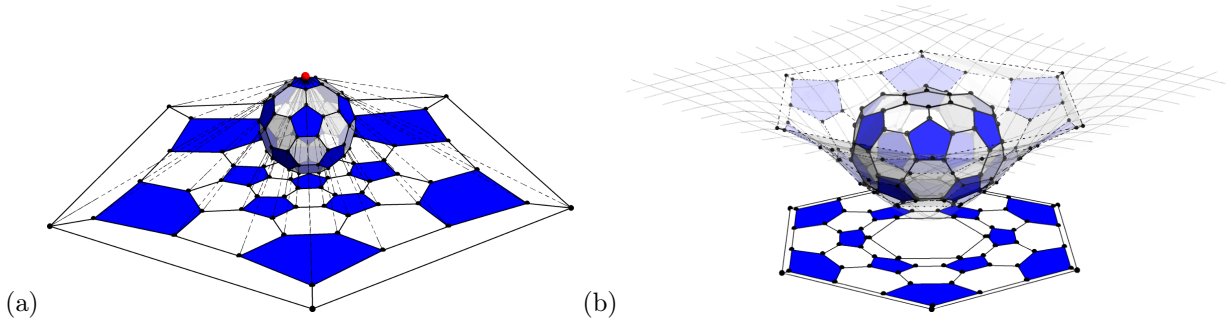


Figure 7: a) Perspective Schlegel projection and b) cone projection for C_{60}

serves for a first rough picture. Furthermore, for large fullerenes it becomes critical to correctly set the projection point or beginning point of the cone. If, for example the projection point is too far away from the fullerene, edges in the 2D graph may cross.

3) Tutte 2D embedding with linear scaling (2D-TE-LS). Here the Tutte algorithm is used [13] but the resulting overcrowding of small polygons near the center of the graph is remedied by a linear scaling procedure

$$\lambda_i = 1 + \frac{f(r_{\min} - r_i)}{r_{\min}} \quad (38)$$

applying a linear scaling factor λ_i to all vertices v_i ($i = 5(6), \dots, n_v$), where f is a parameter to be determined, r_{\min} is the smallest distance to the vertices of the peripheral taken from the barycenter P of the innermost ring (or alternatively the barycenter of the 2D convex hull), and r_i is the distance from that point P to the vertex v_i .

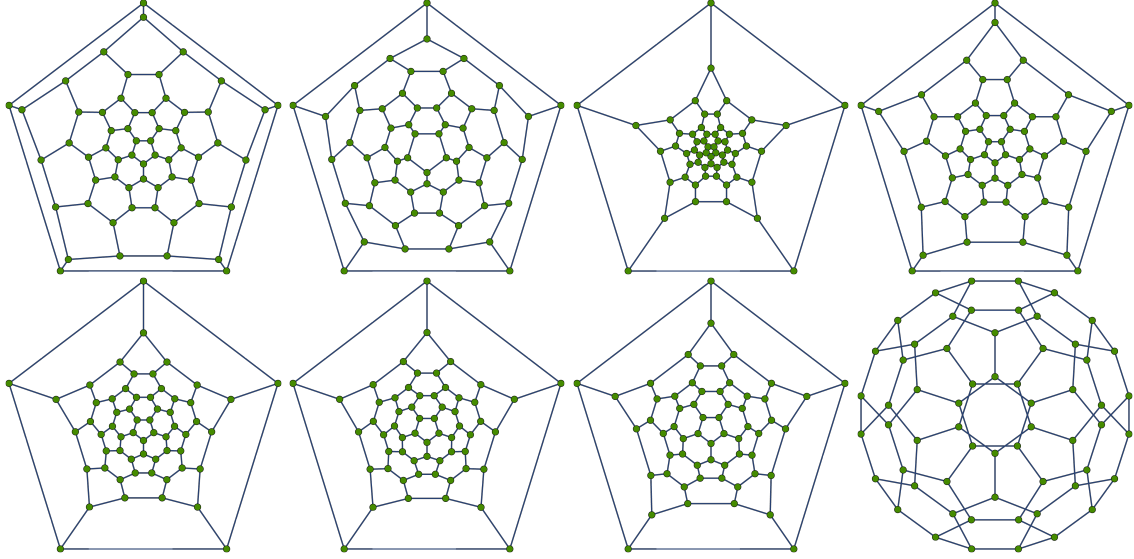


Figure 8: From the left to right: 2D fullerene structures created using **ISchlegel**=1 to 8 described in the input section.

4) Pisanski-Plestenjak-Graovac embedding algorithm (PPGA) [59]. This is one of the more successful algorithms developed by Pisanski, Plestenjak and Graovac in conjunction with a simulated annealing procedure for better visualization of graphs, which is an improvement over usual spring embedders. Here the potential between adjacent vertices is modeled by

$$E_{\text{PPG}} = \sum_{i < j}^{\text{ord}(G)} A_{ij} r_{ij}^2 \exp \left(\alpha \frac{2d_{\max} - d_{iP} - d_{jP}}{d_{\max}} \right) \quad (39)$$

where A_{ij} is the adjacency matrix, r_{ij} the distance between vertex i and j , and α a parameter to be

determined. The distances in eq.(39) are defined through the topological distance matrix D ,

$$d_{\max} = \max_{j,j_P} \{D_{ij_P}\} \quad \text{and} \quad d_{i_P} = \min_{j_P} \{D_{ij_P}\} \quad (40)$$

where j_P is the vertex number belonging to the (5 or 6) peripheral vertices. Instead of a simulated annealing algorithm we start with a Tutte 2D graph and perform a minimization procedure for E_{PPG} .

For other 2D graph embedding methods implemented as shown in Figure 8 see the following input section.

4 Input description

Input and output files are in the directories *input* and *output* respectively. *Program Fullerene* has been tested for many fullerenes found in the following input files:

- 1: All files starting with **CC** have cartesian coordinates for input, for example for C_{20} (**CCc20.inp**), C_{24} (**CCc24.inp**), C_{60} (**CCc60.inp**), or C_{540} (**CCc540.inp**).
- 2: All files starting with **GEOM** use basic geometry to construct C_{20} or C_{60} , e.g. **GEOMc60.inp** or **GEOMc60exp.inp**.
- 3: All files starting with **EDGE** have vertex connectivities as input, e.g. **EDGEc20.inp**.
- 4: All files starting with **RSPI** have the ring-spiral pentagon indices as input, e.g. **RSPIc32.inp** or **RSPIc60NT-D5d.inp** including and the non-spirable fullerenes C_{380} (**RSPIc380.inp**) and C_{384} (**RSPIc384.inp**).
- 5: All files starting with **ISOMER** produces an isomer list, e.g. **ISOMERc78.inp**.
- 6: All files starting with **GC** uses the Goldberg-Coxeter transform of C_{20} , e.g. **GCK2L0c168.inp**.
- 7: All files starting with **READ** uses an external **.xyz** file to read in in cartesian coordinates, e.g. **READxyz.inp**. NB: You need to produce such a file first.

Many more examples can be found. The cartesian coordinates used in the input files are mostly singlet electronic state B3LYP aug-cc-pVDZ optimized up to C_{60} , cc-pVDZ up to C_{180} , and 6-31G for the rest. For some of the fullerenes listed, the singlet state chosen may not be the electronic ground state. Many definitions depend on the use of Ångströms for distances, so please use this unit throughout. A typical input is in *Namelist format*, or if additional data are required in *free format*, and reads like this:

```
C80 Using FM algorithm to produce coordinates for C80 nanotube,
input file: \filename{RSPIc80NT.inp}
&General NA=80 /
&Coord ICart=2, IV2=4, IV3=5 , rspi=1, 2, 3, 4, 5, 6, 37, 38, 39, 40, 41, 42 /
&FFChoice Iopt=1 /
&FFParameters /
&Hamilton IHam=1 /
&Isomers IPR=1 /
&Graph ISchlegel=2, IS01=2, IS02=4, IS03=5 /
```

The first line is a text line, the second **&General** is a general command line in *namelist format* containing many useful flags, the third **&Coord** line tells the program how the coordinates are created, the next two lines **&FFChoice** and **&FFParameters** are specifications for the force field to be used (none in this case), line 5 (**&Hamilton**) concerns the Hamiltonian cycles, line 6 (**&Isomers**) is for an isomer list to be created, and line 7 (**&Graph**) for the 2D fullerene graph (e.g. Schlegel diagram). The last lines in the input is for other input if required, e.g. the indices for vertex insertions (not shown here).

In the following, the input in the required sequence is described in detail (either *Namelist* or in *Free Format*):

4.1 Comment line

A comment line of of max. 132 characters. You can have as many comment lines as you want, so long as they are 132 characters or less (overflow characters do not appear in the output): every line until the first namelist line is interpreted as a comment.

4.2 &General line

Namelist line that contains the main flags for the program.

List of options: **NA**, **IP**, **ixyz**, **ichk**, **nohueckel**, **loop**, **ipsphere**, **nosort**, **ispsearch**, **novolume**, **TolR**, **R5**, **R6**, **filenamedb**, **filename**, **ndbconvert**, **IPMC**.

Option description:

NA Number of atoms (vertices) N_A (Default: 60)

IP If set to 1, verbose output is be produced, i.e. the full distance matrix, all Hamiltonian cycles and all ring connections up to three rings (Default: 0). Warnings: Usage of IP=1 with Iham not set to zero (see below) is a dangerous option and should be used only for small fullerenes.

iwext Flag for producing input file for *CYLview*, *Avogadro* or other plotting programs in standard formats (Default: 0) Default name for the external file is **Fullerene-3D** with the appropriate ending. Otherwise **filename**='filename' needs to be specified (max. 50 characters). Note that if the routine which writes out .xyz, .cc1 or .mol2 files is used more than once, for example through the **nloop** parameter, a number is added after the string "-3D" in order to not cause conflict with previous files. Note that the filenames will have a number attached to it when written out, i.e. **filename-3D.xyz**, **filename-3D2.xyz** etc. (the case **nloop**=1 is omitted in the name)

If **iwext** = 1: Write out external .xyz file named **filename-3D.xyz** for further input.

If **iwext** = 2: Write out external .cc1 file named **filename-3D.cc1** for further input.

If **iwext** = 3: Write out external .mol file named **filename-3D.mol** for further input.

irext Flag for reading external input file which contains cartesian coordinates or vertex connectivities (Default: 0). Note this makes the input of **ICart** obsolete if set non-zero. If **irext** = 1: Read .xyz from file specified by **filename**. The final name is chosen as **filename.xyz** (Default: **Fullerene.xyz**). Another .xyz file is also produced and named **filename-3D.xyz** if **iwext** = 1 is specified.

If **irext** = 2: Read .cc1 from file specified by **filename**. The final name is chosen as **filename.cc1** (Default: **Fullerene.cc1**). Another .cc1 file is also produced and named **filename-3D.cc1** if **iwext** = 2 is specified. .cc1 files contain information about vertex connections. Programs like PY-MOL can read such files. This is useful for reading for example .cc1 files produced by M. Yoshida [60], which has been added to our database.

If **irext** = 3: Read .mol from file specified by **filename**. The final name is chosen as **filename.mol** (Default: **Fullerene.mol**). Another .mol file is also produced and named **filename-3D.mol** if **iwext** = 3 is specified. The file is in standard Tripos format.

nohueckel If **nohueckel** = 1 diagonalization of Hückel matrix is avoided which is of order $\mathcal{O}(n^3)$ (recommended for large matrices over size 5000) (Default: 0).

loop If **loop** = 1: New input required (compound job), i.e. program does not stop at the end but loops back to the beginning (Default: 0). This is important if one takes an initial structure and subjects it to further transformations, for example in subsequent Stone-Wales or Goldberg-Coxeter transformations. You might use this option to read from a .xyz file created in the previous run.

If **loop** = 2: Same as above but if **ICart** specified correctly, it takes pentagon indices from a previous run.

ipsphere Project all vertices onto the minimum covering sphere (MCS), useful for drawing 2D fullerene graphs (Default: 0).

1: Project vertices onto MCS.

2: In addition, write MCS to the .xyz-file `filename-3DMCS.xyz`.

3: In addition, write MCS to the .cc1-file `filename-3DMCS.cc1`.

nosort If **nosort** = 1, permutation of vertices for optimizing Z-Matrix is suppressed (Default: 0).

IPMC If **IPMC** = 1, the number of perfect matchings are counted using the Pfaffian (Default: 0).

ispsearch If **ispsearch** = 0, search for the canonical ring spiral is avoided (Default: 1). If **ispsearch** = 1, ring spiral search takes place until it is successful. If **ispsearch** = 2, all spirals are counted (this is computationally expensive for very large fullerenes).

novolume If **novolume** = 1, all parts for volume determination is avoided (Default: 0).

TolR Tolerance in % (Default: 33). Only change this parameter if the program cannot produce correctly the adjacency matrix. This is only necessary for the cartesian coordinate input (**ICart**=1) and only fails if bond distances are unusually large or small. Connectivities are found for atoms with distances between **R6**= r_{min} and **R6** * (1 + **TolR**/100). If this parameter is set at a value too large, unwanted connectivities are produced resulting in smaller polygons. This parameter should reflect the maximum deviation in distance from the smallest distance found.

filename Prefix for for all external filenames created by the program. Limited to 50 characters. Default: Fullerene.

filenameout Prefix for for a database filename to be used as output for .xyz and .cc1 files if different from **filename**. Limited to 50 characters, Used for example to read from the Yoshida or House of Graph files with **filename** in the directory `database/Yoshida` or `database/HOG`. and write .xyz-file out to another filename called **filenameout**.

ndbconvert Programmers option to convert output from isomer routine into a more compact format.

imcs By setting **imcs**=1 only minimum covering sphere is calculated. This works only for **ICart**=1 (see next input section), however, in this case any cartesian coordinate input can be taken, i.e., any points in 3D space.

itop If **itop** = 1, after creating the adjacency matrix skips directly to the topological analysis. Good for very large fullerenes, e.g. Goldberg-Coxeter transforms. If **itop** = 2 in addition writes out the connectivity vector to external file **ic3file** (formatted), which can be used as further input.

4.3 &Coord line

Input to create cartesian coordinates for the program (e.g. `&Coord ICart=2, R6=1.42 /`).

List of options:

ICart, **rspi**, **jumps**, **IV1**, **IV2**, **IV3**, **isonum**, **leap**, **IGCtrans**, **kGC**, **lGC**, **ISW**, **KE**, **IYF**, **ISW**, **mirror**, **IPRC**, **R5**, **R6**, **ScaleRad**, **nanotube**

Option description:

ICart Flag for the construction of the 3D structure (cartesian coordinates) (Default: 0). For more details see below.

If **ICart** = 1: Cartesian Coordinate input is expected.

If **ICart** = 2, 3: in this case 12 pentagon indices are used for input using **rspi**.

If **ICart** = 4, 5: Goldberg-Coxeter transformation of C_{20} is used, i.e. $GC_{k,l}(C_{20})$ with $k \geq l$ and $k > 0$.

If **ICart** = 2 or 3 and **isonum** \neq 0: pentagon indices are taken from the isomer list contained in a database (see below).

If **ICart** = 6 or 7 input vertex connectivities (edges) directly. If **isonum** \neq 0, then the isomer number from the isomer list contained in the "House of Graphs" database is taken. For the vertex connectivities input as series of cards is required each with 4 integers: **IV1**, **IC1**, **IC2**, **IC3**, i.e., 1 3 6 4 gives the following edges: 1-3, 1-6, and 1-4. Zeros for **IC2** and **IC3** can be used, i.e. 1 3 6 0 gives 1-3 and 1-6, and 1 3 0 0 just gives 1-3 as an edge. For the last card in the input stream, **IV1** needs to

be set to zero to indicate the end of input. For an example see input file `EDGEc60.inp`. It is legal but not necessary to define same edges twice.

If **ICart** = 8 or 9, general spiral input is required using **rspi** and **jumps** (if required).

If **ICart** = 10 A spiral search algorithm is used, see **isearch** keyword in **Isomer** section.

When importing graphs from the “House of Graphs” database, the isomers are numbered starting at 1 (Although, internally we start counting at 0.). There are two example files called `DBc50H0G.inp` and `DBc384H0G.inp`.

rspi An array of 12 comma or blank separated pentagon indices. **rspi** can be given in case of **ICart**={2,3,8,9}. The integers are of the form $n_1n_2\cdots n_{12}$, where the n_i are the 12 Fowler-Manolopoulos ring-spiral pentagon indices, which uniquely identify the locations of the pentagons if the fullerene is spirable [9]. Use the canonical pentagon ring indices if possible (however, a non-canonical from should work as well).

nanotube Flag for the construction of the smallest D_{5h} and D_{5d} fullerene nanotubes, or for the somewhat larger D_{6h} and D_{6d} fullerene nanotubes. The $D_{5h/d}$ nanotubes have caps on each side consisting of 6 fused pentagons, while the $D_{6h/d}$ nanotubes have 6 pentagons connected to the top and bottom hexagon. This option creates the **rspi** as defined above just from the vertex number. Note that for the $D_{5h/d}$ nanotubes one must have fullerenes of the type C_N with $N = 20 + 10n$, and for $D_{6h/d}$ we have $N = 24 + 12n$.

If **nanotube** = 1 Create graph from **rspi** (not needed in input) for $D_{5h/d}$ nanotubes.

If **nanotube** = 2 Create graph from **rspi** (not needed in input) for $D_{6h/d}$ nanotubes.

jumps An array of up to 10 jump positions j_i and step lengths l_i , e.g., $(j_1, l_1, j_2, l_2, \dots, j_5, l_5)$. **jumps** is required in case of **icart**={8,9} (unless you use the general spiral algorithm to create a graph that does not require jumps [which is possible of course]). j_i denotes the face to jump to, and l_i the offset of the cyclic shift ($l_i = 1$ is the smallest possible value, a jump $l_i = 0$ doesn't have any effect).

IPRC There are two databases, one for the general isomers (**IPRC** = 0), and one for the IPR isomers (**IPRC** = 1), the definition is similar to the IPR parameter below (Default: 0).

IV1 Position number for Hückel P-type eigenvector for AME algorithm (Default: 2). The position number is taken from the sequence of Hückel eigenvalues (see ref. [9]).

IV2 Position number for Hückel P-type eigenvector for AME algorithm (Default: 3)

IV3 Position number for Hückel P-type eigenvector for AME algorithm (Default: 4)

isonum Isomer number according to the scheme introduced in Fowler and Manolopoulos' book [9] (Default: 0).

leap If **leap** = n_{leap} the initial fullerene C_{N_A} is subjected to an n^{th} leapfrog transformation. This converts the number of atoms N_A to $N'_A = 3^{n_{\text{leap}}} N_A$. (Default: 0)

ICGtrans If **IGCtrans** = 1 the initial fullerene structure is subjected to a Goldberg-Coxeter transformation $GC_{k,l}[C_{N_A}]$ (Default: 0). In this case **kGC** and **IGC** in the namelist input have to be specified. For $l = 0$ (halma transformation), $k = 1$ and $l = 1$ (leapfrog transformation), and for the Goldberg-Coxeter transformation of C_{20} ($k \geq 1, l \geq 0$) more efficient and faster algorithms are implemented compared to the general case.

ISW If **ISW** = 1 the initial fullerene structure is subjected to a Stone-Wales transformation (Default: 0). In this case an input is required (after all the other input, last card) with pairs of pentagon ring numbers. Numbers are between 1 and 12. These can be obtained from a previous output in the section where ring connections are analyzed and Stone-Wales patterns are printed. In the output for Stone-Wales numbers the first and last ring numbers are the ones required as these are the pentagons. Alternatively you can set the second number to zero, the first number now determines that the Nth Stone-Wales pattern found in the list is taken. For example, an input

2 3 5 0

means pentagon 2 and 3 are taken for the first Stone-Wales transformation, and the 5th in the printed list of Stone-Wales patterns for the second one (see output).

KE If **KE** = 1 the initial fullerene structure is subjected to a Endo-Kroto 2-vertex insertion (Default: 0). In this case an input is required (after all the other input, last card) with pairs of pentagon ring numbers. Numbers are between 1 and 12. These can be obtained from a previous output in the section where ring connections are analyzed and Endo-Kroto patterns are printed. Input is equivalent to the Stone-Wales transformation described above.

IYF If **IYF** = 1 or 2 then a Yoshida-Fowler 4-vertex insertion [16] is performed (Default: 0). In this case an input is required with either hexagon numbers (**IYF**=1) or the position in the list of Yoshida-Fowler patterns given in the output (**IYF**=2). If **IYF** = 3 or 4 then a Yoshida-Fowler 6-vertex insertion [16] is performed (Default: 0). In this case an input is required with the three hexagon numbers (**IYF**=3) or the position in the list of Yoshida-Fowler patterns given in the output (**IYF**=4).

IBF If **IBF** = 1 the initial fullerene structure is subjected to a 6-vertex insertion (Default: 0). Similar to Yoshida-Fowler, an input is required (see **RSPIC384.inp** for details).

mirror If **mirror** = 1 invert all coordinates (take the mirror image). This may be important for chiral fullerenes to obtain the other enantiomer (Default: 0).

IPRC See entry for **isonum**.

R5 Pentagon bond distance in Ångström, i.e. the bond length of the bonds in pentagons (Default: 1.455).

R6 Hexagon bond distance in Ångström, i.e. the bond length of the bonds in hexagons (Default: 1.391). If **R5** = **R6** is chosen then the ideal capped icosahedron for C₆₀ is obtained if **NA**=60 is chosen.

ScaleRad Scaling parameter for Tutte sphere (Default: 4.0). After mapping the fullerene graph on a unit sphere, the radius of the sphere is scaled up by **ScaleRad**/ \langle average bond length \rangle . More non-spherical structures require the choice of a larger **ScaleRad**.

More details for the ICart-flag: If **ICart** = 0 No coordinate input required, coordinates are constructed for the ideal IPR isomer. Currently only works for C₆₀ and C₂₀. If **ICart** = 1 Cartesian Coordinates expected as input. In this case, **NA** extra lines are required in .xyz-format:

$$\begin{array}{cccc} N_{Z_1}, & X_1, & Y_1, & Z_1 \\ & \vdots & \vdots & \\ N_{Z_{NA}}, & X_{NA}, & Y_{NA}, & Z_{NA} \end{array}$$

Here, X_i, Y_i, Z_i are the cartesian coordinates for atom i in Å. N_{Z_i} is not used by this program, but is kept such that .xyz-files can be simply copy/pasted from other quantum theoretical programs to the input file. If **ICart** = 2 or 3 the adjacency matrix is created from a ring-spiral pentagon list.

If **ICart** = 4 or 5 adjacency matrix is created from a Goldberg-Coxeter transform of C₂₀, i.e. $GC_{k,l}[C_{20}]$ with $k \geq l$. In this case the parameter **kGC** in the namelist input has to be used.

If **ICart** = 2, 4, 6 or 8 the AME algorithm using P -type eigenvectors produced from the adjacency matrix to get cartesian coordinates. If the P -type eigenvectors are not in sequence (position 2,3 and 4, see ref.[9] for details), three integer values **IV1**, **IV2**, **IV3** can be specified in the **&Coord** namelist input identifying the position of the eigenvectors (see **RSPIC80NT.inp** for such an example). In this case a warning occurs which means you should carefully check the eigenvectors used and cartesian coordinates produced. Otherwise coordinates produced are useless. This is more often the case as you might expect.

If **ICart** = 3, 5, 7 or 9 the Tutte embedding (3D-TE) algorithm is chosen for the construction of the 3D fullerene structure from the adjacency matrix. This should always work, although the fullerene created might be too spherical compared to the AME algorithm. But this algorithm is easier, and (in theory) should never fail. Examples are given in the input files starting with "pentagon". Problems may arise with very large fullerenes ($n_v > 20000$). This is known to us and we are working on it.

4.4 &FFChoice and &FFParameters lines

Options for force-field optimization.

List of options: **&FFChoice** has parameters **Iopt**, **ftol**, **ihessian**, **iprinth**.
&FFParameters has parameters **fCoulomb**, **WuR5**, **WuR6**, **WuA5**, **WuA6**, **WufR5**, **WufR6**,
WufA5, **WufA6**, **ExtWuR55**, **ExtWuR56**, **ExtWuR66**, **ExtWuA5**, **ExtWuA6**, **ExtWuDppp**,
ExtWuDhpp, **ExtWuDhph**, **ExtWuDhhh**, **ExtWufR**, **ExtWufA**, **ExtWufD**.

Option description:

Iopt Flag for force-field optimization (Default: 0)

If **Iopt** = 1 then fullerene is optimized using the force field method of Wu et al. [20] within a Fletcher-Reeves-Polak-Ribiere algorithm to find the minimum structure. There are more sophisticated force fields available [61, 62], e.g. *Avogadro* has a more sophisticated force-field which you can use, but the Wu force field does the job to create good initial cartesian coordinates for further refinement using more sophisticated QM methods. A converged energy E much greater than zero (i.e. $E > 0.5$) implies that the set distances and angles in the Wu force field cannot be reached for all atoms and rings.

If **Iopt** = 2 Preoptimize with input force field, then optimize with standard Wu force field. This is especially useful for the **fCoulomb** input (see below).

Iopt = 3 triggers an extension of the Wu force field. In addition to Wu (two different bond types and two angle types) there is a third bond type and four types of dihedral angles. If **Iopt** = 3 is chosen, the structure typically converges in less cycles.

Iopt = 4 behaves like **Iopt** = 3, but there is an additional repulsive coulomb force applied. It is switched off as soon as the gradient falls below a value of 10.

ihessian = 1 then Hessian matrix is calculated and diagonalized. The $3n_v - 6$ eigenvalues are sorted according to their values and degeneracies and should be greater than zero for a minimum, and the last six should be close to zero representing translation and rotation. The frequencies produced are used to calculate the zero-point vibrational energy of the fullerene. **iprinth** = 1: print all eigenvalue information. **iprinth** = 2: print all eigenvalue information and full Hessian matrix.

ftol The convergence tolerance on the energy (Default: 5.0×10^{-7} kJ/mol)

WuR5, **WuR6**, **WuA5**, **WuA6**, **WufR5**, **WufR6**, **WufA** These are force-field parameters for the Wu force field, i.e. for distances R5, R6, and angles A5, A6. The force constants are: distance WufR5, WufR6, WufA5 and angles WufA6 (see paper by Wu et al. for details [20]). Defaults: **WuR5** = 1.455, **WuR6** = 1.391 (note these two distances differ from the original Wu paper), **WuA5** = 1.08d2, **WuA6** = 1.2d2, **WufR5** = 1.0d6, **WufR6** = 1.1d6, **WufA5** = **WufA6** = 1.d5. If **fCoulomb** > 0.0, then add an additional repulsive Coulomb force from the barycenter to all atoms (Default: **fCoulomb** = 0.d0). This is extremely useful for an initial geometry optimization to keep the fullerene cage convex if for example the Tutte construction is not a good guess for the initial structure. In such cases, **fCoulomb** = 100.0 is a good choice. Note that some fullerene structures are difficult to optimize if the start is bad, and one needs to play with the **fCoulomb** parameter (see for example RSPIC380.inp).

4.5 &Hamilton line

Option for calculating Hamiltonian cycles and IUPAC numbers.

List of options: **IHam**, **IUPAC**, **ihamstore**

Option description:

IHam

If **IHam** > 0 Then Subroutine HAMILTON is called (Default: 0).

If **IHam** = 1 Routine will stop after 1 million Hamiltonian cycles are found.

If **IHam** = n with $n > 1$ and $n < 10$ then the number of Hamiltonian cycles is 10^{IHam} . **IHam** = 9 basically means that the program runs forever and prints if IHam is reached (dangerous choice).

If **IP** = 1 in the **&General**-input, all Hamiltonian cycles are printed.

If **ihamstore** = 1 the vertex numbers in the cycles are stored as an external file called **filename.ham** (in I3 Format).

IUPAC

If **IUPAC** = 0, the program goes into a fast subroutine and prints only the total number of Hamiltonian cycles. If **IUPAC** = 1, IUPAC numbers are produced (Default: 0).

If set, and **IP** = 0 in the **&Coord**-input, the best Hamiltonian cycle is printed. (See Babić, [49]).

4.6 &Isomers line

Option for producing list of isomers and properties. If available, data are taken from database provided on our website.

List of options: **IPR**, **isearch**, **IPH**, **IStop**, **Ichk** (Default 0 for all these options), **isomerl** (Default 1), **isomerh** (Default to very large value)

Option description:

IPR If **IPR** > 0 then the ring-spiral subroutine of Fowler and Manolopoulos is used [9]. This sub-program catalogues fullerenes with a given number of vertices using the spiral algorithm and a uniqueness test based on equivalent spirals. The required input is IPR.

If **IPR** = 1 for isolated-pentagon isomers from C_{60} onwards.

If **IPR** = 2 for general isomers (this generates a large output and takes some computer time for fullerenes larger than C_{100}). Also the database has all this information (see below) up to C_{100} .

If **isearch** = m ($m > 0$) then an input of initial RSPIs is required, otherwise the RSPi of the closest icosahedral fullerene C_n with $n \leq n_v$ is taken (Default 0). The input is connected to the IPR input otherwise the default value of **IPR** = 1 is chosen. This option searches for all possible isomers in the neighbourhood of $\{i_p - m, i_p, i_p + m\}$. Note that **isearch** = 2 is relatively fast, larger values can soon become computer time expensive.

If **IPH** = 1 then number of distinct Hamiltonian cycles is calculated for every isomer (this is computer time extensive). Note that if this parameter is set but the database does not contain the number of Hamiltonian cycles, it will start to produce them, which is very computer time consuming.

If **istop** = 1 program stops after calling this routine.

If **Ichk** = 1 Restart: Isomer list is continued from previous output file called 'checkpoint' as default if not otherwise give in **filename.chkpnt**. This is a restart option from a previous run which terminated. The new output file does not contain the previous one. The resulting output is a catalogue of the isomers found containing their idealized point groups, canonical spirals, and NMR patterns. If **isomerl** = m , then start printing from isomer list in database from isomer m . If **isomerh** = k , then end printing from isomer list in database at isomer k . [9].

NB: The Brinkmann algorithm used for the House of Graphs is much faster and we are currently implementing it in our program [5].

4.7 &Graph line

Option for producing (X_i, Y_i) coordinates for fullerene 2D graphs (called Schlegel diagrams by some authors).

List of options:

ISchlegel, **ISO1**, **ISO2**, **ISO3**, **IFS**, **nhamcyc**, **PS**, **Scale**, **ScalePPG**, **boost**, **ndual**, **labelvert** (Default: 0 for all values). We recommend **ISchlegel** = 1, 2, 4 or 7.

Option description:

ISchlegel If **ISchlegel** > 0: Use the subroutine Graph2D for generating planar drawings of the Fullerene graph. The value of **ISchlegel** determines the method that is used:

- 1: Use the perspective projection method (PSP), also called Schlegel projection.
- 2: Use the cone projection method (CSP).
- 3: Produce the Tutte embedding (2D-TE).
- 4: Produce the Tutte embedding and perform linear scaling (2D-TE-LS) (scale factor can be read in).
- 5: Produce the Tutte embedding and perform spring embedding optimization $(r_{ij} - r_0)^2$ with r_0 set to 2.0 (2D-SE).
- 6: Starting from the Tutte embedding, perform spring + repulsive Coulomb with prefactor of 0.3 for the force embedding optimization (2D-SE+C). The repulsive force is taken from the barycenter to the vertex. r_0 is set to 2.0. The force constants are set such that the graph looks nice (if you are lucky).
- 7: Starting from the Tutte embedding, perform a Pisanski-Plestenjak-Graovac embedding (PPGE) optimization.
- 8: Starting from the Tutte embedding, perform a Kamada-Kawai embedding [63] optimization using the distance matrix (2D-KKE). This gives a 2D picture of a 3D structure, thus it is not related to a Schlegel diagram and has edge crossings.

ISO1, ISO2, ISO3

If **ISO1** = 0: Use the input coordinates for the construction of the Schlegel diagram.

If **ISO1** \neq 0: Specifying the ring center, edge or vertex through which the z -axis goes at the top of the projection. This rotates the fullerene around the origin of points into the right position for the Schlegel diagram. Since 3 atoms uniquely define the ring, two the edge, and one the vertex, the input is three integers with the corresponding atom numbers (**ISO1**, **ISO2**, and **ISO3**), i.e. for these values (1, 0, 0) is a vertex with atom 1 on the top of the z -axis; (7, 8, 0) is an edge between atom 7 and 8 and z -axis goes the middle of the bond; (12, 13, 50) is a ring (either pentagon or hexagon determined by the program) and z -axis goes through the center of the ring. You can run the program first with (0, 0, 0), check the positions and run it again with the appropriate values. For **ISchlegel** = 1 the input (if chosen) requires to be a ring, i.e. **ISO1**, **ISO2**, and **ISO3** are required, otherwise they are chosen by the program using the largest z -coordinate.

Scale Scaling factor f_{Scale} for linear scaling of Tutte graph (default 2.5). 2D coordinates are scaled by $1 + 0.5f_{Scale}(r_{min} - r)/r_{min}$. r is the distance of the vertex from the barycenter, r_{min} is the smallest distance of the vertex from the barycenter belonging to the outer circumference pentagon or hexagon.

ScalePPG Scaling factor for exponential in Pisanski-Plestenjak-Graovac algorithm [59] (default 1.0).

boost Extra boost for vertices in circumferencing (outer) ring in cone projection method (default 1.2).

PS If **PS** \neq 0: Projection angle in degrees is chosen (default 45 degrees) to be used as an input parameter in the cone projection method. The angle is the cone angle from the point of projection to the projection plane, which touches the point with the smallest z -value (opposite the projection point). Angle is reset to 45 degrees if chosen larger than 89 degrees. In the case of the perspective projection **PS** is the distance between the focal point and the ring center underneath. This is chosen by the program, but if nonzero this parameter has to be carefully chosen. The picture produced gives a good idea if **PS** was correctly chosen or not. The ring centers get an extra boost of 10% in the scaling factors such that they appear more in the center of the rings produced by the Schlegel projection.

IFS

If **IFS** = 1: A tex file named `filename-2D.tex` is produced which contains the 2D fullerene graph (Default: 0).

If **IFS** = 2: A data file named `filename-2D.dat` is produced which contains all the information for producing a 2D fullerene graph with a plotting program.

If **IFS** = 3: Both files are produced.

ndual If **ndual** = 1, The 2D representation of the dual graph is drawn as well in tex-file `filename-2D.tex`. (Default: 0).

labelvert If **labelvert** = 1, The vertices are labelled by number. (Default: 0).

If **nhamcyc** = k then the k-th Hamiltonian cycle produces in the subroutine HAMILTON is taken for use in the 2D graph (Default: 0).

The different algorithms for drawing 2D fullerene graphs are shown in Figure 8. Note that for very large fullerenes which are far from spherical symmetry, the algorithms here may not produce the best 2D graph representation. We are currently working on this problem.

5 Fullerene Isomer Database

A database is provided for general isomers up to C_{120} and for IPR isomers up to C_{122} including the number of Hamiltonian cycles, and without Hamiltonian cycles up to C_{150} for general and up to C_{200} for IPR isomers. The database should be copied into the main program folder and can be used to read the ring-spiral pentagon indices. The files are compressed (gzip) so you need to uncompress the individual file used (gunzip) before usage. The numbering scheme is in canonical order and identical to the one introduced in the book by Fowler and Manolopoulos [9], that is each isomer in the book's appendix can be constructed easily from our database. Examples are given in input files `DBc50.inp` and `DBc66.inp`. Note that these files require the `c40all.database` and `c66all.database` in the `database/ALL` folder. The data files are formatted and can easily be read by the program. It is our intention to extend the isomer list beyond C_{150}/C_{200} (without Hamiltonian cycles) using the "House of Graphs" database. New lists will be available on our website in due time. The determination of the number of distinct Hamiltonian cycles is NP-complete and beyond 100 (120 for IPR) computationally very demanding. The longest file for our database ran for 1 month on 400 CPUs. The Yoshida database (from <http://www.jcrystal.com/steffenweber/gallery/Fullerenes/Fullerenes.html>) often used has also been added into our database and is placed in the folder `database/Yoshida` [60]. The input file `DBc180Yoshida.inp` shows an example for reading from this database. Finally, files from the "House of Graphs" database can be added into the `database/HOG` directory (they need to be downloaded from the House of Graphs website <http://hog.grinvin.org/Fullerenes>) and read-in. An input example is `DBc384HOG.inp`. The directory database needs to be in the same directory as "source" or "libgraph". Note that the maximum number of vertices in the database is defined in the `config.f` file (`LimitALL` and `LimitIPR`).

Bibliography

- [1] Program VMD, "Visual Molecular Dynamics", <http://www.ks.uiuc.edu/Research/vmd/>.
- [2] Brinkmann, G., McKay, B. D., *Discrete Math.*, **2005**, 301, 147-163.
- [3] G. Brinkmann, "Program Fullgen - a program for generating nonisomorphic fullerenes", see <http://cs.anu.edu.au/bdm/plantri/>.
- [4] G. Brinkmann, J. Goedgebeur, and B. McKay, "Program Buckygen - a program for the efficient generation of all nonisomorphic fullerenes", see <http://caagt.ugent.be/buckygen/>.
- [5] G. Brinkmann, J. Goedgebeur, and B. McKay, "The Generation of Fullerenes", *J. Chem. Inf. Sci.* **52**, 2910-2918 (2012).
- [6] G. Brinkmann, O. D.Friedrichs, S. Liskens, A. Peeters, N. Van Cleemput, "CaGe - a Virtual Environment for Studying Some Special Classes of Plane Graphs - an Update", *Match Commun. Math. Comput. Chem.* **2010**, 63, 533-552; see <http://www.math.uni-bielefeld.de/~CaGe/>.
- [7] T. Pisanski, "Program VEGA - A Mathematica based system for manipulating graphs", see <http://www.ijp.si/>.
- [8] W. Myrvold, B. Bultena, S. Daugherty, B. Debroni, S. Girn, M. Minchenko, J. Woodcock, and P.W. Fowler, "FuiGui: A Graphical User Interface for Investigating Conjectures about Fullerenes", *MATCH Commun. Math. Comput. Chem.* **58** 403-422 (2007).
- [9] P. W. Fowler and D. E. Manolopoulos, "An Atlas of Fullerenes" (Dover Publ., New York, 2006).
- [10] R. Tonner, G. Frenking, M. Lein, and P. Schwerdtfeger, "Packed to the Rafters - Filling up C₆₀ with Rare Gas Atoms", *Chem. Phys. Chem.* **12**, 2081-2084 (2011).
- [11] P. Schwerdtfeger, L. Wirz, J. Avery, "The Topology of Fullerenes", *WIRE Comput. Mol. Sci.* **5**, 96-145 (2015).
- [12] P. Schwerdtfeger, L. Wirz, J. Avery, "Fullerene - A Software Package for Constructing and Analyzing Structures of Regular Fullerenes", *J. Comput. Chem.* **34**, 1508-1526 (2013).
- [13] W. T. Tutte, "How to Draw a Graph", *Proc. London Math. Soc.* **13**, 743-767 (1963).
- [14] J. Avery, to be published (2015).
- [15] M. Endo and H.W. Kroto, "Formation of carbon nanofibers", *J. Phys. Chem.* **96**, 6941-6944 (1992).
- [16] M. Yoshida and P. Fowler, "Systematic relationships between fullerenes without spirals", *Chem. Phys. Lett.* **278**, 256-261(1997).
- [17] G. Brinkmann and P. W. Fowler, "A catalogue of growth transformations of fullerene polyhedra", *J. Chem. Inf. Comput. Sci.* **43**, 1837-1843 (2003).
- [18] A. J. Stone and D. J. Wales, "Formation of carbon nanofibers", *Chem. Phys. Lett.* **128**, 501-503 (1986).
- [19] W. T. Vetterling, and B. P. Flannery, in "Numerical Recipes in C - The Art of Scientific Computing", chapter 10, section 6: Conjugate Gradient Methods in Multidimensions; W. H. Press, S. A. Teukolsky (Editors), Cambridge University Press; 2nd edition (1992).
- [20] Z. C. Wu, D. A. Jelski, and T. F. George, "Vibrational Motions of Buckminsterfullerene", *Chem. Phys. Lett.* **137**, 291-295 (1987).
- [21] L. N. Wirz, R. Sure, R. Tonner, A. Hermann, and P. Schwerdtfeger, "A Harmonic Force-Field Method for Fullerenes and a Comparison to Density Functional Calculations for Goldberg-Coxeter Fullerenes up to C₉₈₀", *J. Comput. Chem.*, DOI: 10.1002/jcc.23894 (2015).
- [22] B. Plestenjak, "An algorithm for drawing Schlegel diagrams", see <http://www-lp.fmf.uni-lj.si/plestenjak/Papers/NICEGR.pdf>.

- [23] J. Cioslowski, N. Rao, and D. Moncrieff, "Standard Enthalpies of Formation of Fullerenes and Their Dependence on Structural Motifs", *J. Am. Chem. Soc.* **122**, 8265-8270 (2000).
- [24] D. Babić and O. Ori, "Matching polynomial and topological resonance energy of C_{70} ", *Chem. Phys. Lett.* **234**, 240-244 (1995).
- [25] D. Babić, "Topological Resonance Energy of Fullerenes", *J. Chem. Inf. Comput. Sci.* **37**, 920-923 (1997).
- [26] M. Alcamí, G. Sanchez, S. Diaz-Tendero, Y. Wang, and F. Martin, "Structural Patterns in Fullerenes Showing Adjacent Pentagons: C_{20} to C_{72} ", *J. Nanosci. Nanotech.* **7**, 1329-1338 (2007).
- [27] H. Wiener, "Structural Determination of Paraffin Boiling Points", *J. Am. Chem. Soc.*, **69**, 17-20 (1947).
- [28] A. T. Balaban, "Topological indices based on topological distances in molecular graphs", *Pure Appl. Chem.*, **55**, 199-206 (1983).
- [29] G. Brinkmann, J. Goedgebeur, H. Mélot, and K. Coolsaet, *Discrete Applied Mathematics*, **161**, 311-314 (2013). House of Graphs: a database of interesting graphs, available for fullerenes at <https://hog.grinvin.org/Fullerenes>.
- [30] C. Y. Legault, "Program CYLview", Université de Sherbrook, Canada. See <http://www.cylview.org>.
- [31] Program Avogadro, "An advanced molecule editor and visualizer designed for cross-platform use in computational chemistry, molecular modeling, bioinformatics, materials science, and related areas", see <http://avogadro.openmolecules.net>.
- [32] Program Jmol, "A Java-based development component", see <http://jmol.sourceforge.net/>.
- [33] Program Pymol, "A user-sponsored molecular visualization system on an open-source foundation", see <http://www.pymol.org/>.
- [34] A. T. Gabriel, T. Meyer, and G. Germano, "Molecular graphics of convex body fluids", *J. Chem. Theory Comput.* **4**, 468-476 (2008). The program is available at <http://qmga.sourceforge.net/>.
- [35] F. Cataldo, A. Graovac, and O. Ori "The Mathematics and Topology of Fullerenes", Springer, Berlin (2011).
- [36] P. W. Fowler, G. Caporossi, and P. Hansen, "Distance Matrices, Wiener Indices, and Related Invariants of Fullerenes", *J. Phys. Chem. A* **105**, 6232-6242 (2001).
- [37] D. Cvetković, P. Fowler, P. Rowlinson, and D. Stevanović, "Constructing fullerene graphs from eigenvalues and angles", *Linear Algebra Appl.* **356**, 37-56 (2002).
- [38] M. Jerrum, "Counting, sampling and integrating: algorithms and complexity", Springer, Berlin (2003).
- [39] P. W. Fowler and K. M. Rogers, "Spiral Codes and Goldberg Representations of Icosahedral Fullerenes and Octahedral Analogues", *J. Chem. Inf. Comput. Sci.*, **41**, 108-111 (2001).
- [40] E. Estrada, "Characterization of 3D molecular structure", *Chem. Phys. Lett.*, **319**, 713-718 (2000).
- [41] T. Doslić, "Bipartivity of fullerene graphs and stability", *Chem. Phys. Lett.*, **412**, 336-340 (2005).
- [42] A. T. Balaban, D. Mills, O. Ivanciuc, and S. Basak, "Reverse Wiener Indices", *Croat. Chem. Acta* **73**, 923-941 (2000).
- [43] A. Heydari and B. Taeri, "Szeged index of $TUC_4C_8(S)$ nanotubes", *Europ. J. Combinatorics* **30**, 1134-1141 (2009).
- [44] D. Vukicevic, F. Cataldo, O. Ori, and A. Graovac, "Topological efficiency of C_{66} fullerene", *Chem. Phys. Lett.* **501**, 442-445 (2011).
- [45] P. W. Fowler, "Systematics of fullerenes and related clusters", *Phil. Trans. R. Soc. Lond. A* **343**, 39-52 (1993).
- [46] O. Ori, private communication (2012).

- [47] T. Réti, I. László, "On the Combinatorial Characterization of Fullerene Graphs", *Acta Polytechnica Hungarica*, **6**, 85-93 (2009).
- [48] E. Albertazzi, C. Domene, P. W. Fowler, T. Heine, G. Seifert, C. Van Alsenoy, and F. Zerbetto, "Pentagon Adjacency as a Determinant of Fullerene Stability", *Phys. Chem. Chem. Phys.* **1**, 2913-2918 (1999).
- [49] D. Babić, "Nomenclature and Coding of Fullerenes", *J. Chem. Inf. Comput. Sci.* **35**, 515-526 (1995).
- [50] L. Wirz, D. Babić, J. Avery, and P. Schwerdtfeger, "Toward Tight Upper and Lower Bounds for Hamiltonian Cycle Counts in Fullerene Graphs", in preparation.
- [51] "Numerical Recipes in Fortran 77. The Art of Scientific Computing" (2nd Edition, 1992); see <http://www.nr.com/>.
- [52] M. Yoshida and P. W. Fowler, "Dihedral fullerenes of threefold symmetry with and without face spirals", *J. Chem. Soc., Faraday Trans.* **93**, 3289-3294 (1997).
- [53] L. N. Wirz, and P. Schwerdtfeger, "A general face-spiral algorithm for planar cubic graphs and applications to non-spiralable polyhedra", in preparation.
- [54] S. Díaz-Tendero, F. Martín, and M. Alcamí, "Structure and Electronic Properties of Fullerenes C_{52}^{q+} : Is C_{52}^{2+} an Exception to the Pentagon Adjacency Penalty Rule?", *ChemPhysChem* **6**, 92-100 (2005).
- [55] P. W. Fowler, T. Heine, and F. Zerbetto, "Competition between Even and Odd Fullerenes: C_{118} , C_{119} , and C_{120} ", *J. Phys. Chem.* **104**, 9625-9629 (2000).
- [56] E. A. Yildirim, "Two Algorithms for the Minimum Enclosing Ball Problem", *SIAM Journal on Optimization*, **19**, 1368-1391 (2008)
- [57] T. H. Hopp and C. P. Reeve, "An Algorithm for Computing the Minimum Covering Sphere in Any Dimension", NIST, US Department of Commerce (1996).
- [58] P. A. Heiney, J. E. Fischer, A. R. McGhie, W. J. Romanow, A. M. Denenstein, J. P. McCauley Jr., A. B. Smith, and D. E. Cox, "Orientational ordering transition in solid C_{60} ", *Phys. Rev. Lett.* **66**, 2911-2914 (1991).
- [59] T. Pisanski, B. Plestenjak, and A. Graovac, "Nice Program and its Application, *Croat. Chem. Acta* **68**, 283-292 (1995).
- [60] M. Yoshida, "VRML gallery of Fullerenes" (1991). The database is available at <http://www.jcrystal.com/steffenweber/gallery/Fullerenes/Fullerenes.html>.
- [61] A. Ceulemans, B. C. Titeca, L. F. Chibotaru, I. Vos, and P. W. Fowler, "Complete bond force fields for trivalent and deltahedral cages: Group theory and applications to cubane, closo-dodecaborane, and buckminsterfullerene", *J. Phys. Chem. A* **105**, 8284-8295 (2001).
- [62] I. D. Hands, J. L. Dunn, and C. A. Bates, "A complete nearest-neighbor force field model for C_{60} ", *J. Chem. Phys.* **120**, 6912-6921 (2004).
- [63] T. Kamada and S. Kawai, "An Algorithm for Drawing General Undirected Graphs", *Inform. Process. Lett.* **31**, 7-15 (1989).