Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

GOPY: A tool for building 2D graphene-based computational models

Sebastian Muraru [a,*], Jorge S. Burns [a], Mariana Ionita [a,b]

[a] Faculty of Medical Engineering, University Politehnica of Bucharest, Gh Polizu 1-7, 011061 Bucharest, Romania
[b] Advanced Polymer Materials Group, University Politehnica of Bucharest, Gh Polizu 1-7, 011061 Bucharest, Romania

A R T I C L E I N F O

Article history:
Received 25 March 2020
Received in revised form 30 August 2020
Accepted 1 September 2020

Keywords:
Graphene
Graphene derivatives
Computational models
PDB

A B S T R A C T

GOPY is a free and open-source Python tool specifically written to automate the generation of 2D graphene-based molecular models such as pristine graphene (PG) and several graphene derivatives i.e. graphene oxide (GO), reduced graphene oxide (rGO), aminated polyethylene glycol functionalised reduced graphene oxide (rGO-PEG-NH_2), and N-doped graphene (NG) in the Protein Data Bank file format (PDB). These models are generally built manually, but the process can become lengthy and cumbersome. That is especially the case when investigating larger molecules such as those used in Molecular Dynamics (MD) simulations. Using GOPY significantly speeds up the process from hours to minutes, reducing potential bias that may come with the manual placement of functional groups on a graphene layer. Moreover, the building procedure becomes effortless for the researcher, granting the possibility of producing larger and more complex molecular models than one would be able to build manually. Of its more intensive tasks, the generation of a 4 x 4 nm^2 rGO-PEG-NH_2 layer takes about 9 min on a CodeOcean capsule. Each model is generated in the PDB format, which is easily convertible to a wide array of other molecular formats.

Code metadata

Current Code version	*1.0*
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX_2020_144
Legal Code License	*GPL*
Code Versioning system used	*none*
Software Code Language used	*Python 3.6*
Compilation requirements, Operating environments & dependencies	*Numpy, Scipy*
If available Link to developer documentation/manual	https://github.com/Iourarum/GOPY/blob/master/README.md
Support email for questions	sebmuraru@gmail.com

Software metadata

Current software version	*1.0*
Permanent link to executables of this version	https://github.com/Iourarum/GOPY
Legal Software License	*GPL*
Computing platform/Operating System	*Microsoft Windows, Linux, OS X*
Installation requirements & dependencies	*Numpy, Scipy (Python 3.6)*
If available Link to user manual — if formally published include a reference to the publication in the reference list	https://github.com/Iourarum/GOPY/blob/master/README.md
Support email for questions	sebmuraru@gmail.com

1. Motivation and significance

Graphene, among novel 2D materials, exhibits exceptional electronic, magnetic, optical, mechanical and thermal properties [1–6]. This is essentially due to its unique 2D sp^2 hybridised

[*] Corresponding author.
E-mail addresses: sebmuraru@gmail.com (S. Muraru), jpjsburns@gmail.com (J.S. Burns), mariana.ionita@polimi.it (M. Ionita).

https://doi.org/10.1016/j.softx.2020.100586

network of carbon atoms arranged in a honeycomb lattice. The addition of different functional groups on its surface, the creation of holes or the doping of the graphene layer are some modifications that can critically alter the properties of a graphenic sheet [7]. All of these aspects make graphene and its derivatives useful in a wide array of scenarios [7,8].

Besides being intensely used in wet laboratory research, graphene is receiving significant attention in computational studies, which complement experimental techniques. *In silico* approaches, such as Molecular Dynamics (MD) simulations, are used to provide unique perspectives unachievable through experimental means alone. They allow researchers to study the interactions occurring between molecules of interest at a nanoscale viewpoint. This aspect makes both the design and testing of molecular systems more accessible when compared to setting up laboratory experiments, while also allowing for finer tuning. Resulting insights can highlight what may otherwise remain undetectable in an experimental setting and improve laboratory protocols.

Graphene has been involved in a multitude of MD studies in diverse areas such as graphene-based biosensors [9–12], separation membranes [13–18], hybrid materials [19,20] and investigations of other phenomena [21–29]. A necessary step before running any MD simulation involves building the molecular models of interest. Thus far, we are not aware of an existing solution for automatically generating models of graphene-based 2D materials beyond pristine graphene (PG), e.g. the NanoTube plugin of VMD [30]. We assume most computational models of graphene oxide (GO), reduced graphene oxide (rGO) and other forms of graphene used in computational research were built manually using software similar to Avogadro [31]. Unfortunately, manually adding functional groups to a graphene-layer is a time-consuming process, subject to potential bias and unsuitable for the generation of larger and more complex models. Considering these limitations, we have developed GOPY, which has been used in our previous work [12] to generate graphene-based molecular models, and made it available at https://github.com/Iourarum/GOPY. GOPY can be used to generate files of pristine graphene, graphene oxide, reduced graphene oxide, aminated polyethylene glycol functionalised reduced graphene oxide (rGO-PEG-NH_2) and N-doped graphene (NG) in the Protein Data Bank file format (PDB). We foresee our tool providing a smarter alternative to manually building graphene-based computational models.

2. Software description

2.1. Software architecture

Overview

GOPY is written in Python 3 and makes use of the scipy and numpy libraries. The tool can be used to generate one of the following five molecular models: pristine graphene, graphene oxide, reduced graphene oxide, aminated polyethylene glycol functionalised reduced graphene oxide and N-doped graphene. The main logical approach behind each of the five different functionalities is pivoted on two classes, called Atom() and Typical_Bond(), and the *identify_bonds* function.

Atom() class: Each object instance of this class contains the usual information stored in a PDB file, thus the following instance attributes: atom_number, atom_name, residue_name, residue_number and the X, Y and Z coordinates of the atom.

Typical_Bond() class: Each object instance of this class has a length and an identity instance attributes. Object instances are supposed to describe a bond that can form between two atomic species in terms of the bond's typical length. The identity attribute is made from the corresponding atom and residue names of the two species and helps in selecting appropriate distances to determine whether a bond is formed, based on the atomic species involved.

Importing a PDB file: GOPY allows one to import a PDB file containing either a PG or a GO layer through the *read_in_graphene* and *read_in_GO* functions. Formatting of the PDB files should respect the guidelines shown in Fig. 1: atoms making up the graphene layer are expected to be named "CX" or "CY" and belong to a residue named "GGG". Similarly, when importing a GO layer, atoms belonging to carboxyl, epoxy and hydroxyl groups should also respect the naming rules shown in Fig. 1 and belong to residues named "C1A" for carboxyl, "E1A" for epoxy and "H1A" for hydroxyl groups. If the PG layer was generated using VMD, GOPY will by default consider "C ␣ ␣ ␣ GRA ␣ X" to be "CX ␣ ␣ GGG ␣ ␣". Only the recognised atoms are going to be imported and become new object instances of the Atom class.

The *identify_bonds* function describes the main approach used in determining whether placement of new atoms, such as when adding functional groups to the graphenic surface, can be considered valid or is rather incorrect and must be dropped. Given an Atom() object, together with a list containing all Atom() objects, the function is initially used to compile a list of all the neighbouring atoms within a certain Euclidean distance larger than all values of the Typical_Bond() objects. The bond that may form between the Atom() object given as input and each of its neighbours is then compared to existing Typical_Bond() objects, looking both at the identity and length attributes. Should these match, the bond is considered 'identified' and is appended to a list returned by the function. In GOPY, new atoms are placed one by one and thus the *identify_bonds* function is essentially expected to return a list containing only one element for a placement to be considered successful. The only case where two elements are expected is at the addition of epoxy groups.

To aid in the process of better understanding and using GOPY, the user may seek a quick introduction by typing "python GOPY.py help" or look through the comments in our code available at: https://github.com/Iourarum/GOPY. Furthermore, both flowcharts (Supplementary Figure 1) and function charts (Supplementary Figure 2) are available as supplementary materials.

Pristine Graphene Generation

Pristine graphene represents a single layer of sp^2 carbon atoms arranged in a honeycomb lattice [4]. Relying on the function *generate_pristine_graphene*, GOPY provides the option of generating a PG layer given as input the desired X and Y dimensions. The algorithm behind this feature is rather straightforward, generating a set of coordinates corresponding to the carbon atoms, which are later written to a PDB file. The bond length between carbon atoms is considered fixed at 1.418 Åand all atoms will be placed at coordinate 0.000 on the Z-axis. Therefore, as shown in Fig. 2, on the X-axis, a new carbon atom is found every 1.228 Å, while on the Y axis it is found either after 0.709 Åor 1.418 Å. By dividing the desired length on the X-axis by 1.228 Åand rounding the result, one obtains the number of "halves" of a hexagonal ring necessary to reach it. Similarly, the desired Y-length is divided by 2.127 Å, the sum of 0.709 Åand 1.418 Å, to find the number of complete hexagons that can be placed vertically. Thus, when coordinates of one atom in the graphene sheet are known, the positions of all other atoms should be easily determinable.

The function used to generate the sheet, *generate_pristine_graphene*, relies on two other functions: *fill_hexagon* and *fill_row*. Given the number of hexagon "halves", the latter repeatedly calls on *fill_hexagon* to fill up one row. The function *fill_hexagon* is used to return the coordinates of the vertices of a regular hexagon given the coordinates of one atom.

Regardless of the values offered as input, the first row is placed without the atom numbered 6 as a first step (for numbering of

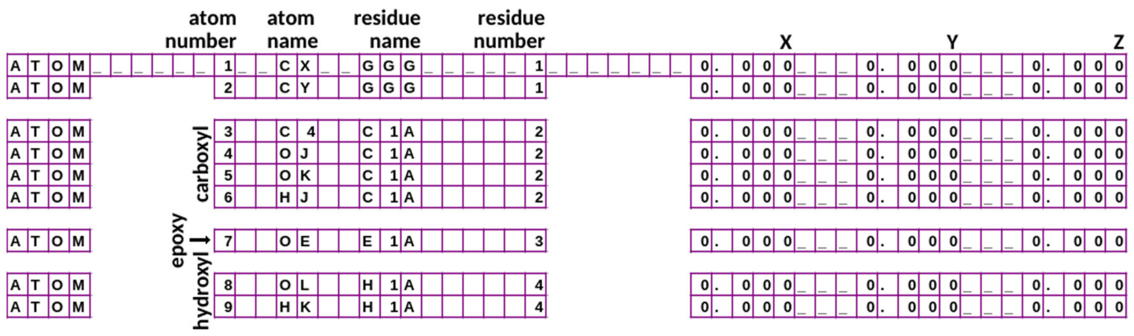
atom number
atom name
residue name
residue number
X
Y
Z
ATOM 1 CX GGG 1 0.000 0.000 0.000
ATOM 2 CY GGG 1 0.000 0.000 0.000
carboxyl
ATOM 3 C4 C1A 2 0.000 0.000 0.000
ATOM 4 OJ C1A 2 0.000 0.000 0.000
ATOM 5 OK C1A 2 0.000 0.000 0.000
ATOM 6 HJ C1A 2 0.000 0.000 0.000
epoxy
ATOM 7 OE E1A 3 0.000 0.000 0.000
hydroxyl
ATOM 8 OL H1A 4 0.000 0.000 0.000
ATOM 9 HK H1A 4 0.000 0.000 0.000

Fig. 1. Formatting of the expected PDB files that GOPY can use as input, the expected atom and residue names in the case of PG and GO layers containing carboxyl, epoxy and hydroxyl functional groups.

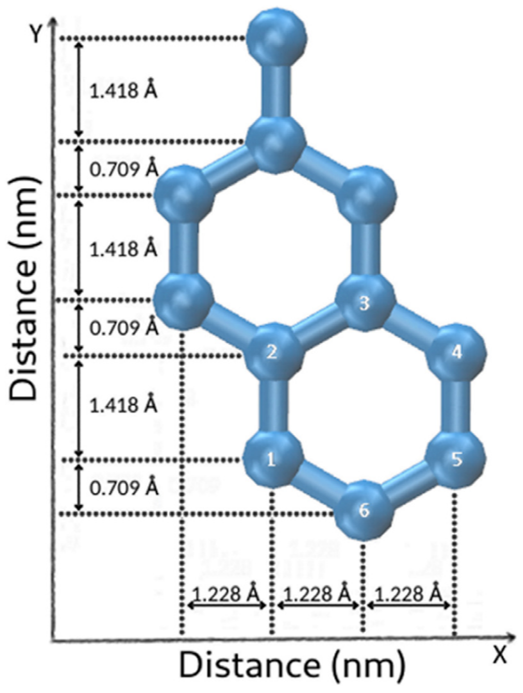

Fig. 2. Distances observed in a PG layer on the X and Y axis, together with the numbering of the carbon atoms in a hexagonal ring. In GOPY, all carbon atoms in a PG layer are placed at coordinate 0.000 on the *Z*-axis.

atoms see Fig. 2). For each row added on top, atoms numbered 2, 3 and 4 become the ones numbered 1, 5 and 6 on the additional, new row. Each new row adds 2.127 Åin height (Y-axis) to the existing structure and thus all complete hexagons that can be placed vertically will be placed as a second step, filling up each row to the desired length.

At the end, by dividing the input value of Y to 2.127 Å, the value of the decimals is analysed: if closer to 1, an entire new row is added; if closer to 0.66 a new row is added without its atom number 3 or if closer to 0.33, atom number 6 is added on the first row. Thus, following these steps, the dimensions of the generated layer will be as close as possible to the ones given as input.

Graphene Oxide Generation

Graphene oxide represents chemically modified graphene with extensive oxidative modification of the basal plane resulting in a structure comprising a large number of oxygen functionalities, including carboxyl (-COOH), hydroxyl (-OH) and epoxide (-O-) functional groups, with a C/O atomic ratio subject to the method of synthesis [32].

Generation of a GO model relies on the *create_GO* function. The function requires as input the path to a PDB file, describing a PG layer, and the desired number of carboxyl, epoxy and hydroxyl groups to be added. Addition of the functional groups on the PG layers is performed using a simple, geometrical approach. This technique is dependent on known bond lengths, stored as objects in the Typical_Bond() class and known molecular angles formed between the atoms of the functional groups. The atoms are added randomly, either above or below the PG layer, one at a time. Placement is considered successful if a newly added atom is found to have only one bond, determined distance-wise using the *identify_bonds* function. The epoxy group is an exception, as two bonds are expected for successful placement. Carboxyl groups are only placed on the carbon atoms at the edges of the PG layer, which are supposed to have at most two detectable bonds and should not already be connected to a functional group. Hydroxyl and epoxy groups are placed everywhere on the PG layer. The epoxy group requires two available carbon atoms, the second being chosen randomly out of those bonded to the first.

For illustrative purposes, placement of a (-COOH) or (-OH) functional group follows the following steps: **(1)** initially, a functional group is chosen from a list containing all functional groups that still have to be added; **(2)** a carbon atom on the graphene layer is picked randomly from those compatible with the selected functional group; **(3)** the first atom of the functional group is added according to typical bond lengths and molecular angles; a test on whether the new atom forms only one valid bond with the carbon atom picked at step (2) is performed using the *identify_bonds* function; if false, everything goes back to step (2) for a maximum of 50 times; **(4)** the rest of the atoms of the functional group are added in a similar manner, one by one; **(5)** when all atoms of a functional group have been successfully placed then those atoms are added to the list of Atom() objects. Placement of an epoxy group is executed following similar steps, though it requires two valid bonds to two carbon atoms that also form a bond between themselves.

The functional groups, the distances between the corresponding atoms and a 3 nm x 3 nm generated GO layer are shown in Fig. 3.

Aminated Polyethylene Glycol Functionalised Reduced Graphene Oxide Generation

Aminated polyethylene glycol-functionalised reduced graphene oxide is a form of functionalisation that introduces a PEG spacer to keep the amino group away from the GO surface and alter physiochemical properties [33]. Obtaining a rGO-PEG-NH_2 molecule involves reduction of GO, partly restoring properties of pristine graphene by removing oxygen-containing groups from its surface [34].

Using GOPY, chains with the formula –NH-$(C_2H_4O)_2$-C_2H_4-NH_2 can be placed as functional groups on a graphenic layer. In this

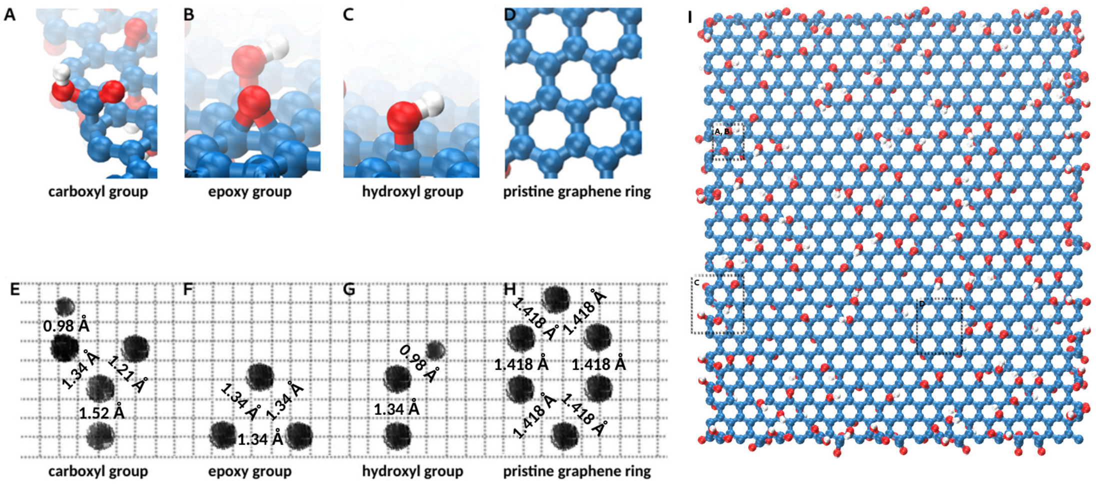

Fig. 3. 3D visualisations of **A**. carboxyl group **B**. epoxy group **C**. hydroxyl group and **D**. graphene ring. Schematic representations and corresponding inter-atomic distances of **E**. carboxyl group, **F**. epoxy group, **G**. hydroxyl group and **H**. graphene ring; **I**. 3D visualisation of a 3 x 3 nm^2 GO layer and the corresponding regions from which A.-D. were generated.

scenario, these molecules are 25 atoms long and display a linear structure. Thus, using the same placing strategy as in the case of GO, based purely on known bond lengths and angles, was found suboptimal in terms of accuracy.

Given that this graphene derivative is obtained in the laboratory starting off with a GO molecule, the main function *add_NH_PEG_NH*$_2$ relies on a GO layer to be imported before proceeding to the generation of the rGO-PEG-NH_2 molecule. The percentages of carboxyl, epoxy and hydroxyl groups to be removed are required as input. All removed epoxy and hydroxyl groups are replaced with the linear molecules that were mentioned in the first paragraph.

Each atom of the linear molecule is placed one at a time, checking that it forms only one identifiable bond to the atom placed beforehand. To maximise the chances for valid placements and optimise the speed of the process, the following strategy was employed: knowing the already existing atom that the new one is supposed to bind to, and the typical bond length between them, a large number of random points are uniformly distributed on the surface of a sphere. The coordinates of the centre of the sphere are identical to the coordinates of the existing atom and its radius is equal to the typical bond length, see Fig. 4A. This is done through the *hydrogen_coord_gen* function, which places one random point with the coordinates X, Y and Z on the surface of a sphere centred in X_0, Y_0, Z_0 with radius r:

$$\theta U\,(0,\,2\pi)\;cos\alpha U\,(-1,\,1)\,X = X_0 + r \times cos\theta \times sin\alpha Y$$
$$= Y_0 + r \times sin\theta \times sin\alpha Z = Z_0 + r \times cos\alpha$$

One of the many points is picked at random and its coordinates would be used for the new atom. However, in order to prevent it from being too close to nearby atoms, points that are within a certain distance of neighbours, correlated to typical bond lengths, are first removed leading to point distributions as shown in Fig. 4A. The validity of placing the new atom at the same coordinates as the chosen point is tested using the *identify_bonds* function, expecting only one bond. Better placement success rates were achieved when adding the H atoms afterwards of the C, O and N atoms making up the skeleton of the linear molecules. Figs. 4B and C display 3D visualisations of a successfully generated 3 nm x 7.5 nm rGO-PEG-NH_2 molecule, while examples of resulting chain geometries can be seen in Fig. 4D. Visualisations of point distributions on a sphere were drawn using Mayavi [35]. The strategy described in this case can be improved by considering parameters such as bond and dihedral angles, which are commonly used in an energy minimisation step in MD, further cropping down the initial point distribution. With the current implementation, GOPY can generate a 4 x 4 nm^2 rGO-PEG-NH_2 molecule with 106 chains of 25-atoms in about 9 min on a CodeOcean capsule.

In a typical run, the first atom of the linear chain, called N2, is placed following the steps: **(1)** a carbon atom previously connected to a functional group is chosen at random; **(2)** all carbon atoms bound to the atom from step (1) are found and placed in a list; **(3)** a large number of random points are distributed uniformly on the surface of a sphere, with its centre at the atom from step (1) and its radius defined as the typical bond length between the N2 atom and the carbon atom; **(4)** spheres with their centres at the atoms from step (2) and radii defined as the typical bond lengths between N2 and carbon atoms; **(5)** all points placed at (3) situated within the volumes of at least two spheres are removed and a point is picked at random out of those left; **(6)** the atom N2 is placed at this location if the chosen point respects the above or below placement of the removed functional group previously bound to the carbon atom from step (1) and has only one identifiable bond when using the *identify_bonds* function. The rest of the atoms making up the chain are placed in a similar manner.

Hole generation

The function *hole_generation* takes as input the path to a PG layer in the PDB format, the desired number of holes and an interval describing their sizes in terms of the number of atoms to be removed. In addition, one should also specify whether the hole should grow in an uni-directional or multi-directional manner and whether it should be allowed to fuse with other holes or not touch the edges of the layer or other existing holes, as shown in Fig. 5. The size of the hole is chosen randomly from the given interval, while the first atom to be removed in order to create a new hole is also picked randomly. All small pieces of the graphene layer made of less than six atoms that do not form any bonds with the rest of the layer, at the end of the hole generation process, are removed.

Creating a hole in a PG layer involves the following steps: **(1)** a list of all available carbon atoms is generated using the *get_map_anywhere* function; **(2)** a list of the carbon atoms situated on the edges of the graphene layer or existing holes is generated using the *get_contour* function; **(3)** one atom is selected according to the preferences specified in the input: if the holes are allowed to fuse then the atom can be found in the list from

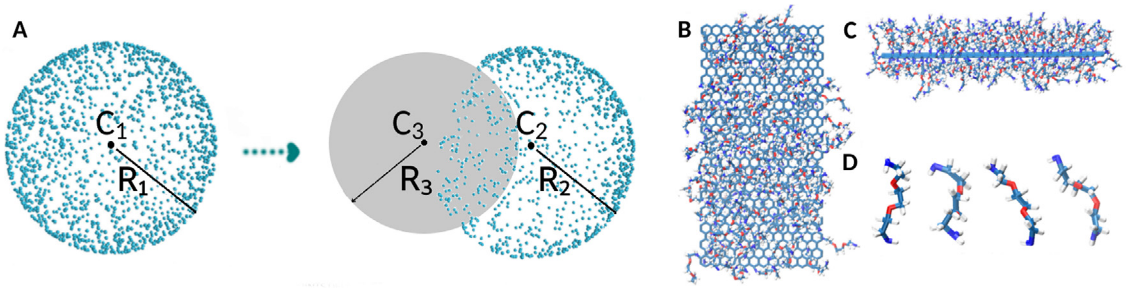

Fig. 4. A. A large number of uniformly distributed random points on the surface of a sphere (C_1, R_1) with its centre at the existing atom and the radius equal to the typical bond length between the existing atom and the new atom that has to be placed. The dotted arrow points to a similar example of uniformly distributed random points on the surface of a sphere (C_2, R_2) with a part of the points removed due to the intersection with another out-of-plane sphere (C_3, R_3) with its centre at a nearby atom and radius equal to the typical bond length between the nearby atom and the one that has to be placed; **B.-C**. 3D visualisations of a generated rGO-PEG-NH_2 layer from different angles; **D**. 3D visualisations of resulting geometries of –NH-(C_2H_4O)-C_2H_4-NH_2 chains.

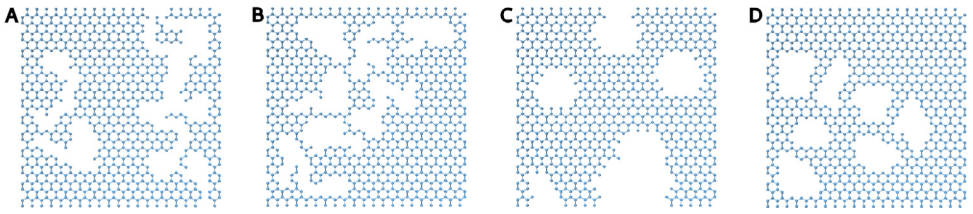

Fig. 5. A.-D. Examples of generated holes in PG layers **A.-B**. Holes were generated by taking out only one neighbour of the most recently removed atom (uni-directional), leading to more linear looking holes; **C.-D**. Holes were generated by taking out multiple neighbours of recently removed atoms (multi-directional), leading to more oval-shaped holes; Holes in A. and C. were allowed to touched edges when growing, while B. and D. were not.

(2), otherwise it should only be found in (1); its neighbours are determined using the *identify_bonds* function; **(4)** growing the hole is done according to the specified criteria: "uni-directional" randomly picks one of the neighbours of the atom from (3); "multi-directional" will pick all available neighbours; "interior" will prevent the removal of atoms found in the list at step (2); "exterior" allows for any atom from the list of neighbours to be picked to enlarge the hole; **(5)** removed atoms become the atom at step (3) and the process is repeated until the required number of atoms is removed.

N-Doped Generation

In N-doped graphene, the spin density and charge distribution of carbon atoms is influenced by the neighbouring nitrogen dopants, which induces the "activation region" on the molecule's surface, thus making it suitable to a different set of applications compared to other forms of graphene [36]. The three bonding configurations that an N atom may adopt on a graphene layer are known as N-Graphitic, N-Pyridinic and N-Pyrrolic [36,37] and can be observed in Fig. 6.

The function *generate_N_doping* requires as input the path to a PG or GO layer in the PDB format and the desired numbers of N-Pyridinic, N-Pyrrolic and N-Graphitic N atoms. The carbon atoms making up the graphene layer and not connected to a functional group are filtered based on the number of bonds they form in the following manner: if forming three bonds then the carbon atom can be replaced by an N-graphitic N atom. Otherwise, if only two bonds are identified then it can be replaced by an N-Pyridinic N atom. In the special case in which one of its neighbours can only form two bonds too, as identified by the *identify_bonds* function, then the atom can be replaced by an N-Pyrrolic N atom. Atoms are then replaced randomly until the desired number of N atoms is reached. N-Graphitic and N-Pyridinic atoms replace the carbon atom at exactly the same spatial coordinates. For N-Pyrrolic atoms, we take into account the centre of geometry of the 6-atom graphene ring and the centre of geometry of the polygon formed by the four atoms left when removing the two with only two bonds each. Placement of the N atom is within 0.3–0.6 Åof the centre of geometry of the 6-atom ring, while the centre of geometry of the 4-atom polygon is useful in determining the direction of placement, away from the 6-atom ring's centre of geometry.

3. Illustrative examples

Illustrative examples of generated atomistic models are shown in Fig. 7. PG layers are shown on the first row, GO layers on the second, rGO-PEG-NH_2 on the third and N-doped layers on the fourth. The dimensions of the graphenic layer in each of the columns are 2 x 2 nm^2, 3 x 3 nm^2, 5 x 5nm^2 and 10 x 10 nm^2 in the last two. The GO layers were generated according to the formula $C_{20}(OH)_2(\text{-O-})_2(COOH)_1$ meaning that for each 20 carbon atoms of the pristine graphene layer, two hydroxyl, two epoxy and one carboxyl groups were added randomly above or below the layer. Carboxyl groups were added only on the edges, while hydroxyl and epoxy groups were distributed throughout the whole surface of the graphenic layer. The rGO-PEG-NH_2 molecules were generated starting off with the GO molecules by removing two thirds of the existing carboxyl groups and replacing all hydroxyl and epoxy groups with the NH-(C_2H_4O)$_2$-C_2H_4-NH_2chains. The resulting seemingly random orientation of the chains is a result of the method described in the manuscript. N-doped graphene molecules were created using the generated PG layers. Some of the molecules presented are shown to have holes. These were generated following PG model generation and then corresponding steps of GO, rGO-PEG-NH_2 and N-doped generation were taken. The molecules are shown for illustrative purposes of the capabilities of the presented software tool, GOPY.

4. Impact

GOPY contributes by providing a simple and easy to use Python implementation for quickly generating computational models of graphene-based 2D molecules in the PDB format. The

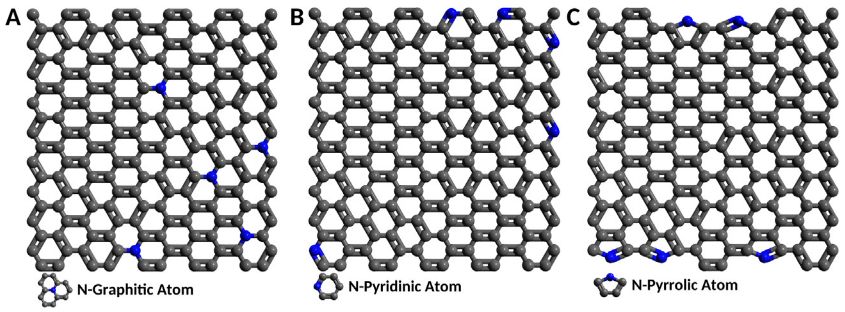

Fig. 6. 3D visualisations of N-doped graphene layers containing **A**. N-Graphitic atoms **B**. N-Pyridinic atoms and **C**. N-Pyrrolic atoms. N atoms are shown in blue.

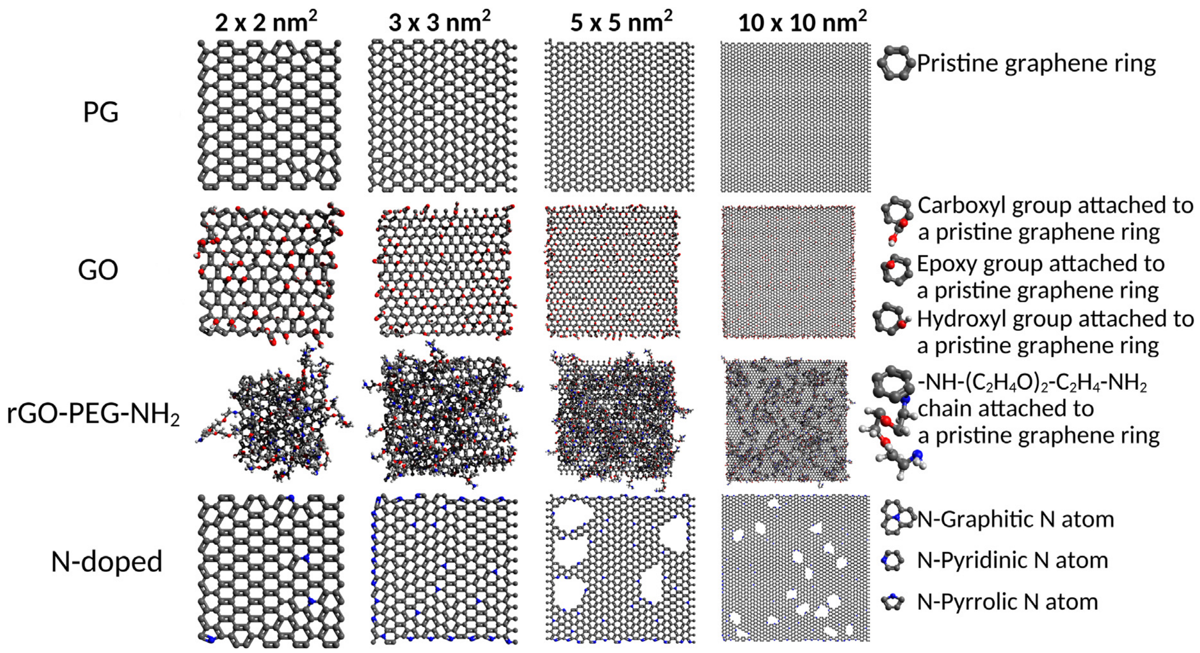
2 x 2 nm²
3 x 3 nm²
5 x 5 nm²
10 x 10 nm²
PG
GO
rGO-PEG-NH₂
N-doped
Pristine graphene ring
Carboxyl group attached to a pristine graphene ring
Epoxy group attached to a pristine graphene ring
Hydroxyl group attached to a pristine graphene ring
-NH-(C₂H₄O)₂-C₂H₄-NH₂ chain attached to a pristine graphene ring
N-Graphitic N atom
N-Pyridinic N atom
N-Pyrrolic N atom

Fig. 7. Set of illustrative examples showcasing generated PG layers on the first row, GO layers on the second, rGO-PEG-NH_2 on the third and N-doped graphene layers on the fourth. The dimensions of the presented layers vary according to the column they were placed in. Some of the layers contain holes generated using GOPY.

format was chosen because of being both widely used and easily convertible into a multitude of other formats such as .mol, .xyz, .gro etc. through tools such as Avogadro [31].

Due to the limited availability of similar tools, we suspect that models used in previous computational chemistry studies, particularly MD, may have been built manually. Building the models manually, however, is time-consuming and potentially subject to bias. On top of that, it is inefficient and may lead one to generally avoid building large and complex systems, as is the case with the rGO-PEG-NH_2 molecule.

Essentially, GOPY can generate PG, GO, rGO, rGO-PEG-NH_2 and N-doped graphene models. For each of its functionalities, the tool allows for plenty of flexibility and can generate a new model in a matter of seconds or up to a few minutes for the larger and more complex scenarios. The algorithms involved, especially the one used for rGO-PEG-NH_2, could be extended to a wide array of molecules meant to act as functional groups on a 2D surface.

We plan on further developing GOPY with the purpose of making it a go-to tool for generating both simple and very complex computational models of 2D materials and their derivatives. The design of a graphical user interface (GUI) for providing even better accessibility to researchers not yet confident in using the command line will also be implemented. As a future functionality, the software may be expanded and include an energy minimisation step, commonly performed in most MD simulations.

Thus, through GOPY we aimed to facilitate the use of large and complex graphene-based molecules by reducing the time spent when building even complicated functionalisation models to minutes, minimising potential bias and replacing the laborious part of the work with a short waiting time when generating multiple models.

5. Conclusions

GOPY is an open source and simple to use command-line tool written in Python, implemented for generating graphene-based molecular models in the PDB format. Using this tool, as opposed to manually building computational models, aims to minimise potential bias and replace the gargantuan task of generating large, multiple and complex models with a short waiting time of a few minutes. Generating multiple, unique models of the desired graphene-derivatives becomes an easily approachable undertaking. We believe GOPY and the underlying molecule-building strategies presented will be of significant help to future computational studies, in particular for those exploring graphene functionalisation and molecular dynamics simulations.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by a grant of the Ministry of Research and Innovation, Operational Program Competitiveness Axis 1—Section E, Program co-financed from European Regional Development Fund under the project number 154/25.11.2016, P_37_221/2015, "A novel graphene biosensor testing osteogenic potency; capturing best stem cell performance for regenerative medicine" (GRABTOP).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.softx.2020.100586.

References

[1] Li P, Wang Z, Qiao Z, Liu Y, Cao X, Li W, et al. Recent developments in membranes for efficient hydrogen purification. J Membr Sci 2015;495:130–68. http://dx.doi.org/10.1016/j.memsci.2015.08.010.
[2] Papageorgiou DG, Kinloch IA, Young RJ. Mechanical properties of graphene and graphene-based nanocomposites. Prog Mater Sci 2017;90:75–127. http://dx.doi.org/10.1016/j.pmatsci.2017.07.004.
[3] Geim AK, Novoselov KS. The rise of graphene. Nature Mater 2007;6:183–91. http://dx.doi.org/10.1038/nmat1849.
[4] Rao CNR, Sood AK, Subrahmanyam KS, Govindaraj A. Graphene: The new two-dimensional nanomaterial. Angew Chem Int Ed 2009;48:7752–77. http://dx.doi.org/10.1002/anie.200901678.
[5] Tiwari SK, Sahoo S, Wang N, Huczko A. Graphene research and their outputs: Status and prospect. J Sci: Adv Mater Dev 2020;5(1):10–29. http://dx.doi.org/10.1016/j.jsamd.2020.01.006.
[6] Almonti D, Ucciardello N. Improvement of thermal properties of micro head engine electroplated by graphene: experimental and thermal simulation. Mater Manuf Process 2019;34(14):1612–9. http://dx.doi.org/10.1080/10426914.2019.1594263.
[7] Huang X, Yin Z, Wu S, Qi X, He Q, Zhang Q, et al. Graphene-based materials: Synthesis, characterization, properties, and applications. Small 2011;7:1876–902. http://dx.doi.org/10.1002/smll.201002009.
[8] Mohan VB, Lau K-T, Hui D, Bhattacharyya D. Graphene-based materials and their composites: A review on production, applications and product limitations. Composites B 2018;142:200–20. http://dx.doi.org/10.1016/j.compositesb.2018.01.013.
[9] Xu Z, Lei X, Tu Y, Tan Z-J, Song B, Fang H. Dynamic cooperation of hydrogen binding and π stacking in ssdna adsorption on graphene oxide. Chem Eur J 2017;23:13100–4. http://dx.doi.org/10.1002/chem.201701733.
[10] Zeng S, Chen L, Wang Y, Chen J. Exploration on the mechanism of DNA adsorption on graphene and graphene oxide via molecular simulations. J Phys D: Appl Phys 2015;48:275402. http://dx.doi.org/10.1088/0022-3727/48/27/275402.
[11] Chen J, Chen L, Wang Y, Chen S. Molecular dynamics simulations of the adsorption of DNA segments onto graphene oxide. J Phys D: Appl Phys 2014;47:505401. http://dx.doi.org/10.1088/0022-3727/47/50/505401.
[12] Muraru S, Samoila CG, Slusanschi EI, Burns JS, Ionita M. Molecular dynamics simulations of DNA adsorption on graphene oxide and reduced graphene oxide-PEG-NH_2 in the presence of Mg^{2+} and Cl^- ions. Coatings 2020;10(3):289. http://dx.doi.org/10.3390/coatings10030289.
[13] Giri AK, Teixeira F, Cordeiro MNDS. Salt separation from water using graphene oxide nanochannels: A molecular dynamics simulation study. Desalination 2019;460:1–14. http://dx.doi.org/10.1016/j.desal.2019.02.014.
[14] Lin H, Gong K, Ying W, Chen D, Zhang J, Yan Y, et al. CO_2 -philic separation membrane: Deep eutectic solvent filled graphene oxide nanoslits. Small 2019;15:1904145. http://dx.doi.org/10.1002/smll.201904145.
[15] Chen Y, Zhu Y, Ruan Y, Zhao N, Liu W, Zhuang W, et al. Molecular insights into multilayer 18-crown-6-like graphene nanopores for K^+/Na^+ separation: A molecular dynamics study. Carbon 2019;144:32–42. http://dx.doi.org/10.1016/j.carbon.2018.11.048.
[16] Liu Q, Wu Y, Wang X, Liu G, Zhu Y, Tu Y, et al. Molecular dynamics simulation of water-ethanol separation through monolayer graphene oxide membranes: Significant role of O/C ratio and pore size. Sep Purif Technol 2019;224:219–26. http://dx.doi.org/10.1016/j.seppur.2019.05.030.
[17] Ye H, Li D, Ye X, Zheng Y, Zhang Z, Zhang H, et al. An adjustable permeation membrane up to the separation for multicomponent gas mixture. Sci Rep 2019;9:7380. http://dx.doi.org/10.1038/s41598-019-43751-0.
[18] Jiao S, Xu Z. Selective gas diffusion in graphene oxides membranes: A molecular dynamics simulations study. ACS Appl Mater Interfaces 2015;7:9052–9. http://dx.doi.org/10.1021/am509048k.
[19] Yu Z, Feng Y, Feng D, Zhang X. Thermal conductance bottleneck of a three-dimensional graphene–CNT hybrid structure: a molecular dynamics simulation. Phys Chem Chem Phys 2020;22:337–43. http://dx.doi.org/10.1039/C9CP05228C.
[20] Rama P, Bhattacharyya AR, Bandyopadhyaya R, Panwar AS. Tunable energy barrier for intercalation of a carbon nanotube into graphene nanosheets: A molecular dynamics study of a hybrid self-assembly. J Phys Chem C 2019;123:1974–86. http://dx.doi.org/10.1021/acs.jpcc.8b10958.
[21] Yang L, Xu H, Liu K, Gao D, Huang Y, Zhou Q, et al. Molecular dynamics simulation on the formation and development of interlayer dislocations in bilayer graphene. Nanotechnology 2020;31:125704. http://dx.doi.org/10.1088/1361-6528/ab5c7e.
[22] Kandezi MK, Lakmehsari MS, Matta CF. Electric field assisted desalination of water using b- and n-doped-graphene sheets: A non-equilibrium molecular dynamics study. J Molecular Liquids 2020;302:112574. http://dx.doi.org/10.1016/j.molliq.2020.112574.
[23] Huang Y-R, Chen C-L, Tseng Y-H, Chang Chien C-T, Liu C-W, Tai C-C, et al. Graphene wrinkles affect electronic transport in nanocomposites: Insight from molecular dynamics simulations. J Mol Graph Model 2019;92:236–42. http://dx.doi.org/10.1016/j.jmgm.2019.07.016.
[24] Poorsargol M, Alimohammadian M, Sohrabi B, Dehestani M. Dispersion of graphene using surfactant mixtures: Experimental and molecular dynamics simulation studies. Appl Surf Sci 2019;464:440–50. http://dx.doi.org/10.1016/j.apsusc.2018.09.042.
[25] Ebrahim-Habibi M-B, Ghobeh M, Mahyari FA, Rafii-Tabar H, Sasanpour P. An investigation into non-covalent functionalization of a single-walled carbon nanotube and a graphene sheet with protein G:A combined experimental and molecular dynamics study. Sci Rep 2019;9:1273. http://dx.doi.org/10.1038/s41598-018-37311-1.
[26] Molla A, Li Y, Mandal B, Kang SG, Hur SH, Chung JS. Selective adsorption of organic dyes on graphene oxide: Theoretical and experimental analysis. Appl Surf Sci 2019;464:170–7. http://dx.doi.org/10.1016/j.apsusc.2018.09.056.
[27] Ai Y, Liu Y, Huo Y, Zhao C, Sun L, Han B, et al. Insights into the adsorption mechanism and dynamic behavior of tetracycline antibiotics on reduced graphene oxide (RGO) and graphene oxide (GO) materials. Environ Sci: Nano 2019;6:3336–48. http://dx.doi.org/10.1039/C9EN00866G.
[28] Yuan R, Ju P, Wu Y, Ji L, Li H, Chen L, et al. Silane-grafted graphene oxide improves wear and corrosion resistance of polyimide matrix: molecular dynamics simulation and experimental analysis. J Mater Sci 2019;54:11069–83. http://dx.doi.org/10.1007/s10853-019-03672-9.
[29] Jin Y, Sun Y, Chen Y, Lei J, Wei G. Molecular dynamics simulations reveal the mechanism of graphene oxide nanosheet inhibition of Aβ_{1--42} peptide aggregation. Phys Chem Chem Phys 2019;21:10981–91. http://dx.doi.org/10.1039/C9CP01803D.
[30] Humphrey W, Dalke A, Schulten K. VMD – VIsual molecular dynamics. J Mol Graph 1996;14:33–8. http://dx.doi.org/10.1016/0263-7855(96)00018-5.
[31] Hanwell MD, Curtis DE, Lonie DC, Vandermeersch T, Zurek E, Hutchison GR. Avogadro: an advanced semantic chemical editor, visualization, and analysis platform. J Cheminform 2012;4:17. http://dx.doi.org/10.1186/1758-2946-4-17.
[32] Chen D, Feng H, Li J. Graphene oxide: Preparation, functionalization, and electrochemical applications. Chem Rev 2012;112(11):6027–53. http://dx.doi.org/10.1021/cr300115g.
[33] Chen J, Liu H, Zhao C, Qin G, Xi G, Li T, et al. One-step reduction and pegylation of graphene oxide for photothermally controlled drug delivery. Biomaterials 2014;35(18):4986–95. http://dx.doi.org/10.1016/j.biomaterials.2014.02.032.
[34] Pei S, Cheng H-M. The reduction of graphene oxide. Carbon 2012;50(9):3210–28. http://dx.doi.org/10.1016/j.carbon.2011.11.010.
[35] Ramachandran P, Varoquaux G. Mayavi: 3D visualization of scientific data. IEEE Comput Sci Eng 2011;13(2):40–51. http://dx.doi.org/10.1109/MCSE.2011.35.
[36] Wang H, Maiyalagan T, Wang X. Review on recent progress in nitrogen-doped graphene: Synthesis, characterization, and its potential applications. ACS Catal 2012;2(5):781–94. http://dx.doi.org/10.1021/cs200652y.
[37] Xu H, Ma L, Jin Z. Nitrogen-doped graphene: Synthesis, characterizations and energy applications. J Energy Chem 2018;27:146–60. http://dx.doi.org/10.1016/j.jechem.2017.12.006.