Visualization of electronic density

Bastien Grosso^{a,b}, Valentino R. Cooper^c, Polina Pine^d, Adham Hashibon^e,
Yuval Yaish^a, Joan Adler^{a,*}

^a*Technion-IIT, Haifa, Israel*

^b*EPFL, Lausanne, Switzerland*

^c*Oakridge National Laboratory, TN, USA*

^d*Loyola University, IL, USA*

^e*Fraunhofer IWM, Freiburg, Germany*

Abstract

The spatial volume occupied by an atom depends on its electronic density. Although this density can only be evaluated exactly for hydrogen-like atoms, there are many excellent algorithms and packages to calculate it numerically for other materials. Three-dimensional visualization of charge density is challenging, especially when several molecular/atomic levels are intertwined in space. In this paper, we explore several approaches to this, including the extension of an analoglyphic stereo visualization application based on the AViz package for hydrogen atoms and simple molecules to larger structures such as nanotubes. We will describe motivations and potential applications of these tools for answering interesting physical questions about nanotube properties.

Keywords: visualization, electronic charge density, nanotube

2010 MSC: 00-01, 99-00

1. Introduction and educational applications

The Computational Physics group at the Technion developed a desktop visualization code for their needs in Atomistic Visualization, called AViz, [1, 2, 3]. It is based on Mesa/OpenGL and Qt. Initially we modelled atoms as balls,

*Corresponding author

Email address: phr76ja@tx.technion.ac.il (Joan Adler)

URL: http://phycomp.technion.ac.il (Joan Adler)

1
2
3
4
5
6
7
8
9
10 spins as cones or vectors and quadrupolar molecules or liquid crystals or pores
11 as cylinders. In a project motivated by educational use we invoked an “off-
12 label” AViz implementation to illustrate the electronic probability density as
13 calculated from the H atom analytic solution in a smoke rendering form [4], us-
14 ing dots to enable semi-transparency. The dot representation of AViz, originally
15 created to enable quick selection of viewing angle etc for atomistic samples, cre-
16 ates a translucent effect whereby the sample’s interiors are visible. Combined
17 with color and rotation it gives excellent insight into the nature of the different
18 electronic states[5].

19
20 In order to draw the electronic density we (obviously) first need to calculate
21 it. In brief, for the H atom one calculates the electronic density on a grid, and
22 defines a box around each grid point. Dots are then drawn at randomly cho-
23 sen points within each box at an average density equal to the local electronic
24 density at the center of the box. Each of these points is given x, y and z coordi-
25 nates and is drawn using the dot feature of AViz. The .xyz format is common to
26 many molecular visualization packages, but its normally used to indicate atoms,
27 not density points. For the hydrogen 2s case the datafiles contain some 50,000
28 points, rather larger than those typically used in atomic visualization, although
29 since they are not solid spheres, the rendering time is reasonable. In the left
30 frames of Figure 1 we show the AViz visualization of the electronic density of
31 the 2s state of the H atom in both greyscale and color. Three dimensional visu-
32 alizations of hydrogen atom wavefunctions are very helpful for teaching Modern
33 Physics or Quantum Mechanics classes. The concept of electronic density is
34 hard to grasp. Animated gifs of these samples in rotation are found at [5] using
35 binned color and have been found to be helpful to students [6].

36
37 Visualization of electronic density for atoms or small molecules is a bit dif-
38 ferent from volume visualization of, for example, fluid flow around an aeroplane
39 wing because in the latter case there are one (or more) solid wings with fluid
40 around, whereas in the atomic case we have one or many pinpoint nuclei and
41 different electron orbitals for each. It is also different than water density in a
42 cloud as electronic densities are more structured. Even the electronic density
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

Figure 1: AViz dot visualizations of the electronic density of the 2s state of an H atom in greyscale (left), and in color (center), and an ethylene molecule in color (right). The atoms in the ethylene are indicated by small squares, and the color scale of the density ranges from orange (most dense) through red and pink to blue (least dense) in all cases.

around a metal has a surface comprised of the inner conduction bands as a platform and a few additional free electrons. As we will mention later on, there has been much study of atomic orbitals with techniques that visualize 2 dimensional cuts, or even sequential planes thereof. Covalent materials such as nanotubes are a bit different than surfaces of metals or even single atoms because one aims for semi-transparency in order to show depth and different layers of density. For our aim of actually measuring the thickness when the surface is curved, we really need to have a 3 dimensional space.

This paper includes detailed references to our codes, and these codes can all be found at [7] and in the CPC program library and server [8]. We provide details with some reference to the Technion's parallel multinode server, TAMNUN, which has the Redhat LINUX operating system and PBS queuing system, but have tested parts on non-parallel LINUX machines with interactive runs as well. The AViz used throughout is version 6.1, which has downloadable sources on [2], and continued development on GitHub [9]. It is currently available for several LINUX flavours, and under development for Windows. In the next section we discuss visualization for molecules and nanostructures, and in Section 3 we describe stereo visualization and binned color smoke rendering. This is followed by a section describing earlier nanotube visualizations, and in Section

5 we present general aspects of the calculations and visualizations. We devote Section 6 to details of the 3D charge density visualization, and in Section 7 some specifics for analyglyphic stereo. Section 8 covers some physics aspects.

2. Molecules and solids, especially nanostructures

A natural extension of electronic density visualization for single atoms is to molecules and simple nanostructures. In this area, experiment has advanced more quickly than simulation. We note that there are many quite standard implementations of smoke density approaches to surface electronic density simulating STM images, but most do not use color as well as concentration to indicate the density of their “smoke”. Nor do they generally publish 3D images which can be rotated and sliced as AViz can, to peer inside the sample. In less transparent visualization some unique aspects of nanostructures may be overlooked.

Our extension of AViz applications aimed to visualize the electronic density resulting from simulations of larger molecules and solids in the same way. Initial studies [10] used a density functional theory approach with Slater-type orbitals (STO-3G) as implemented in GAMESS [11] to examine the electronic densities of simple molecules where there is no analytic solution. The graphical procedure is similar. In the right frame of Figure 1 we show the colored electronic density of ethylene. Note that because the density is not shown with solid curves we can peer nicely into the sample. We have also been able to show methane molecules both with all orbitals and stripped of the lower densities, and have also explored specific orbitals [10, 12, 13].

Our next attempt at electronic density visualization was to periodically bounded samples, employing plane wave DFT calculations as implemented in the Vienna Ab Initio Simulation package VASP[14, 15]. In preparation for the larger samples, we returned to some of the simple molecules with the VASP code. At that time we used slice visualization with VESTA [16], since dot visualization for many atom samples was limited by issues related to the large

datafile size. Despite the VESTA solid visualization, rather than the 3D dot-smoke type, we confirmed that the main features agree. Note that VESTA images also include green in the color range, the early AViz ones only used a red-blue scale for better depth perception.

3. Stereo, binned color smoke rendering

The next stage in our visualization development was to move to 3D stereo. We selected an approach that has a long history, even predating GL. This old concept of anaglyphic stereo relies on two images, slightly displaced, and viewed on a regular screen/projector or poster [17] through colored glasses, or two squares of cellophane. Stereo Vision (SV) works by showing a different image to each eye, thus creating the illusion of a 3D image.

AViz 6.1 [2, 18, 19, 20] has incorporated the possibility of SV, and although more than two colors are possible there remains some color washout, depending on color selection. The SV images generated by AViz, such as those in this paper, are best viewed using red-cyan anaglyphic glasses. The images in Figures 2 show nanotube atoms at two different viewing angles in stereo.

Figure 2: Left: Nanotube side-view. Right: View along the axis of a nanotube.

In Figure 3 we show (on the left) the electron density of an hydrogen atom in stereo. Improved colors for stereo for the H atom, as well as clearer instructions

were given by Meital Kreif in [21, 22]. Two examples are given in Figure 3 (center) and (right), the former for the $n = 3, l = 1, m = 1$ orbital and the latter for the $n = 4, l = 3, m = 2$ orbital of the H atom. On the website [21] all
 105 images can be rotated to further aid in depth perception.

Figure 3: Left: Orbital electronic density of the (3,1,0) state of the hydrogen atom. Center: Image of the $n = 3, l = 1, m = 1$ orbital. Right: Image of the the $n = 4, l = 3, m = 2$ orbital.

4. Motivation and preparation for studying the electronic density of nanotubes

A nanotube vibrates at a frequency that is a function of its width, length, tension, boundary conditions and for certain boundary conditions also of its
 110 type. A molecule placed on such a vibrating tube will change this frequency, enabling elucidation of the mass of the adsorbed molecule. The description of these systems with analytic models is limited in cases when both ends are not completely clamped, as occurs in the laboratory. The essential parameter for model analysis is the width of the nanotube wall, and it is the electronic cloud
 115 around the atomic nuclei that determines this.

In a series of papers and a thesis, Pine and coworkers [23, 24, 25, 26, 27, 28, 29] reviewed the literature and carried out extensive classical molecular dynamics simulations at the atomistic scale to carefully determine the limits of applicability of the analytic theory. While values for the wall width were
 120 deduced indirectly by us and many others, direct estimation is of course more

desirable. It would be even more desirable to automate this. We are interested in the effect of nanotube local distortions (bending, stretching) on the width, and also in the effect of total strain.

The issue of why the nanotube width is important was discussed in [24], but since it is our major physical motivation, we will recapitulate here. For a nanotube of radius r , length L , and volume V the frequency of the n th mode, f_n , of its vibrations depend on the nanotube's width, w and radius r via two quantities used in its calculation: the moment of inertia, $I = \pi r w (4r^2 + w^2)$ and mass linear density $\rho_L = 2\pi r w \rho_V$ (ρ_L is the density per unit length and ρ_V the density per unit volume) which are related to the tubes' Young's modulus, E . The exact relation depends on the boundary conditions and model; and the main point of [24, 25, 26, 27, 28] was to deduce these frequencies without explicit use of any of the competing analytic models. Literature estimates of E range between 1-6 Tpa, and this is known as Yakobsen's paradox [30]. Huang et al. [31] claimed this range is a result of the scattering of width estimates, and proceeds to study this with sheets of graphene and different potentials as well as reporting on some ab initio results. They also mention the importance of boundary conditions and we have discussed factors such as chirality and strain. The width range they quote is between 0.69 nm to 0.0617 nm. In two of our papers [24, 28] we found via two distinct selfconsistent procedures with simulations of nanotubes of multiple lengths and radii that 0.067 nm is consistent with all our simulations and with earlier studies [30, 31]. However we wish to be able to estimate changes in this value directly when we place adsorbed molecules to be weighed, hence the present investigation.

Of course any study to estimate the width directly has to be quantum mechanical. Given the limitations of computer resources, and the need for relatively long tubes, simulated for long times, with a range of parameters it is natural that such width estimates should be deduced with a multiscale approach. Simulations in tandem at an atomistic scale and at an electronic scale are desirable.

The question of multiscale simulations in an efficient manner that minimizes the difficulty of using diverse codes for different scales, inputs and outputs is

currently an important research issue. For example, in the European Union FP7 program several collaborations under the banner of “NMP (Nanosciences, Nanotechnologies, Materials and new Production technologies) multiscale modelling” are researching this issue. In particular the project “Simulation framework for multiscale phenomena in nano and microscaled systems, or SimPhoNy” [32] aims towards a uniform environment for scales from electronic to macroscopic, with visualization at all scales. The present study, in addition to its intrinsic interest provides a prototype test bed for SimPhoNy. Together with the atomistic scale and analytic continuum models developed previously, the present study explores the electronic scale and its visualization development and transfer from atomistic scale. There is the caveat that the wrapper codes described below are C++-based and will have to be moved to python for the SimPhoNy environment.

4.1. Molecular dynamics as the starting point

The nanotube simulations at the atomistic scale that form the basis for the present study are described in [24, 25, 26, 27, 28] with codes given in [29]. In brief the tube is equilibrated with periodic boundary conditions, then “cut open” and clamped with the appropriate boundary conditions. When strain is needed the tube is carefully stretched before clamping. It is then allowed to vibrate for a long time and the vibration spectrum analysed with MATLAB codes given in [29].

We used the Brenner [33] potential, and for the present study we selected four nanotubes with different strains from [28] and used rings near the center of the tubes so as to minimize boundary effects. We show one of the nanotubes with 10 percent strain from [28] in Figure 4. Note the slight extension at the ends to which we will return below. Throughout this study we refer to the axis along the tube as the y direction and the two perpendicular directions as x and z .

Figure 4: Stretched nanotube, the y axis is along the axial direction and x and z directions are perpendicular; observe extension near the ends.

4.2. Earlier nanotube visualizations

For the nanotube simulations we carried out still and animated visualization with AViz, creating .xyz files in the simulations and drawing them post-processing. In addition to valuable help in the debugging phase, we could show, for example, regions where bonds become stretched (or compressed), with an example in Figure 5. This figure as well as the previous one are shown to assist the reader in grasping the local tube geometry. We note that code files that enable direct processing of LAMMPS output into AViz input have also been prepared at the Technion [34]. However for the present project potential problems of explosion of Brenner potential samples in LAMMPS meant that we continued to work with our own older Molecular Dynamics code. The group has also prepared wrappers for outputting Monte Carlo simulations into AViz with both simulations and wrappers in python [35].

4.3. Earlier visualization from electronic density simulations

In addition to our VASP-VESTA studies mentioned above in which we did not succeed to create smoke rendering input, we also carried out exploratory simulations [36] with the Quantum Espresso (QE) [37, 38] DFT code and FORTRAN wrappers for transfer to AViz, which led to the development of the present more substantial study.

5. General aspects of our calculations and visualizations

For this calculation we selected the Quantum Espresso plane-wave DFT code [37, 38]. The selection was based partly on a preference for public domain codes

Figure 5: Vibrating nanotube; upper image shows all nearest neighbour bonds, lower image shows only shorter bonds to emphasise compression regions.

with clear documentation of their format for the charge density and partly on the possibility of more local support.

We illustrate the complete protocol in Figure 6. From the initial molecular dynamics simulation we select one or several rings and place their coordinates in an .xyz file. We confirm their validity with an atomistic AViz vizualization, for a one ring example, see Figure 7. The Quantum Espresso part has two stages: first performing a full DFT structural relaxation (plane wave self-consistent field) with the pw.x [39] subroutine and then executing the post-processing routine pp.x [40] to extract the charge density. We have written C++ wrappers to transform the .xyz coordinates into the input format needed by Quantum Espresso. The latter requires all atoms to be within its simulation box, defined by the cell vectors. One has to translate the coordinates using the C++ code many_rings.cpp in order to have the first atom at the origin and paste the new coordinates in the appropriate section of the input file for pw.x. Another C++ wrapper is then used to convert the output of pp.x into the .xyz for Aviz electronic density visualization. AViz requires an input file in the .xyz format with two initial rows, the first being the integer number of atoms or dots and the second a comment line. All other rows have a letter to indicate atomic type or dot color and at least three real number spatial coordinates.

All files that we use can be found in a single tar file on the website [7] in ddl.charge_density.tar. In addition the input and additional wrapper routines, currently written in C++ perform the tasks listed below. As well as being part

Figure 6: Main steps in the calculation.

of the total download file they are also provided as separate tar files in [7] with the name of the file in brackets in this list:

- charge_density_xyz.cpp [7] extracts the charge from pp.x in the correct input format for AViz (in ddl.example_input.QE.tar)
- many_rings.cpp provides the input coordinates for pw.x (in ddl_many_rings.tar)
- nano4parts.cpp recombines the 4 quarters into a single ring (in ddl_charge_density.tar)
(In the current setup QE initially calculates the density in four quadrants which have to be recombined prior to visualization).

We also recommend that one dilutes the points prior to visualization and this is currently done in this implementation as part of nano4parts.cpp. (This step was not needed in the H atom and simple molecule cases because the approach used to generate points led naturally to a far more dilute concentration.)

We first describe the calculation for a single ring without strain. We then describe the additional stages needed for systems of several rings and for systems under strain. The details of the simulations are given in Appendix A, and details of the transformations on the output to create AViz input files in Appendix B and recombination of the 4 parts in Appendix C.

Figure 7: The one ring system with its long axis, in the y direction, pointing thru its center.

6. Visualization of the charge density in 3D

Once all the steps described above are carried out, we reach the most interesting aspect of this procedure - the visualization of the charge density in 3D. We make an initial coarse graining of the density in order to use the binned color approach using dots. The actual color selection is done interactively within AViz, and can be adapted to the user's preference, optimal selection if there is to be greyscale printing or the user's requirements in the case of a colorblind person. It would also be different if analoglyphic stereo is to be implemented,

because due to the analglyphic “washout” some color palettes are better than others. At this point we have not varied the number of dots in accordance with their local density as was done for the analytic solutions. The color binning is done at the same time as the extraction from the QE output format and all details are in Appendix B. It helps to determine which region of space has a higher probability of electron localization. In Figure 8 we see a straight on view, with colors respectively indicating successively lower densities. The lowest density is not colored for viewing ease, and with colors black, red, green, yellow and blue we show successively higher densities. These very bright colors were selected for their distinctiveness; better palettes for 3D viewing are displayed below. This image is not diluted, the diluted version (details in Appendices) is given in Figure 9.

We observe that there are bad moire effects here. An improved palette for non-stereo viewing and better angular selection is given in two examples of the extended systems, a 3 ring system in Figure 10 and for a 6 ring one in Figure 11.

7. Stereo visualization of electronic density around a nanotube

Further insight into the structure can be found by invoking several advanced AViz features. The fovy (field of view in the y direction) can be tweaked [20] in the panel from the viewpoint button on the AViz main panel (Figure 12) via the explicit option (Figure 13) so that the moire effects of a straight-on cartesian view are minimised.

The analglyphic stereo, as presented in [19] and adjusted by the buttons on the left with the glasses images (Figure 14) gives an image (Figure 15) which when viewed with cyan-red glasses appears to come out of the page. In this figure one can see the higher density (red color) semitransparently behind the green grey lower density even without the glasses. A rotating version of this image is shown on [41].

Figure 8: Final result.

8. Further analysis and conclusions

Having carefully described the computation and visualization of the charge density, we now turn to physical aspects; thus returning to our initial motivations. The first was research into the deformation of nanotube walls under distortions. The present project has demonstrated that we can calculate and visualize the electronic density surrounding the tube. We can observe local variations in its thickness at different axial locations, see for example Figures 10 and 11. Further investigations with larger samples and a careful numerical data

Figure 9: Final result with dilution.

analysis that falls beyond the scope of the present paper have begun and will be extended in the near future to larger and more distorted tubes.

The second motivation concerned the integration of simulations on electronic and atomistic scales as part of SimPhoNy [32]. The present computations provide one stage of efforts towards this goal. We have successfully with initial molecular dynamics input transferred to electronic density functional theory calculations in a smooth, automatic and reproducible manner.

This new visualization technique for viewing electronic density has many potential applications. For example, to the study of any system that involves proximity and contact between different materials such as adsorption of molecules on surfaces. One such case (but certainly not limited to) is the next stage of the nanotube vibration project [27]. Our motivation for the study of nanotube vibrations was to study the way that the adsorption of DNA onto a nanotube would change the vibrational frequencies, hence enabling evaluation of the DNA

Figure 10: Visualization of charge density for a 3-rings system without any strain.

mass. Part of this project also relates [26] to the nature of the nanotube bound-
 300 ary conditions, and in order to mimic the experimental case, where it sits on
 two side supports we need to know how the nanotube settles on these. To de-
 termine these we need to include the electron clouds in our calculations; and
 then observe their structure.

9. Acknowledgements

305 This study is part of the EU "SimPhoNy" collaboration. V.R.C. was sup-
 ported by the U.S. Department of Energy, Office of Science, Basic Energy
 Sciences, Division of Materials Science and Engineering. This research used
 resources of the National Energy Research Scientific Computing Center, sup-

Figure 11: Visualization of charge density for a 6-rings system without any strain

ported by the Office of Science, U.S. Department of Energy under Contract
 No. DEAC02-05CH11231. The main calculations were made on TAMNUN,
 supported by the RBNI Nanotechnology Institute at the Technion. We thank
 Arik Landau for advice on running Quantum Espresso on TAMNUN, Yulia
 Halupovich for her endless support on TAMNUN and Ariella Richardson for
 help with C++.

Appendix A. Details and code links for the simulation part

All tar files are opened with `tar -xvf name_of_file`. The instructions assume
 use of a LINUX system for the compilation and execution (tested on Red Hat),
 and terminal software that enables ssh -X access if the desktop is a different
 system to where the simulations and visualizations are carried out. For com-
 pleteness we also include comments related to batch (PBS) submission on our
 local SGI RedHat Linux parallel cluster called TAMNUN; with the exception
 of commands related to MPI we have also checked their validity on a shared
 memory computer. TAMNUN has Quantum Espresso installed with Intel com-

Figure 12: The viewpoint panel of AViz, the "view" button was pressed and the explicit option selected to bring up the explicit panel.

Figure 13: The explicit button in the viewpoint panel of AViz.

Figure 14: AViz frame with stereo options on the right.

pilers and AViz 6.1 in the /usr/local partition. TAMNUN is a very typical small
 parallel cluster, and its use in the following can be considered generic. The AViz
 used here is the Qt3 6.1 version, but the visualization has been checked on the
 newer AViz 7.0 Qt4 version [9] on UBUNTU. Initial versions of the single ring
 system were run at NERSC on a Cray XT system (hopper).

The following sections contain explanations for sample input files and pa-
 rameter choices; more details are in the full set of downloadable files at [7], and
 non-system specific aspects are in the QE documentation [37, 38]. Three cases -
 single ring, multiple rings and strained nanotubes are considered. For the single
 ring case one downloads the example_input tar file.

Appendix A.1. Input file for pw.x

Figure A.16 shows a typical input file for pw.x. We now present a brief
 discussion of the different lines of the input file; a full description of the doc-
 umentation is linked above. The path to an user directory is /u/username/,
 where username has to be replaced by the login name that is used to connect
 on TAMNUN or the appropriate value for another system. In detail:

Figure 15: A stereo image of the 10 percent strained, fovy adjusted 3 ring nanotube showing the semi-transparent outline of the inner cloud.

340 Appendix A.1.1. $\mathcal{E}control$

- calculation : defines the type of calculation that one wants to do, 'scf' is for self-consistent field.
- pseudo_dir : the path where the potential file(s) are located. This is a user specific choice.
- 345 • prefix : 'NANO' is an example, one can change it and write whatever he/she wants but it is important that all pw.x and pp.x input files use the same prefix.
- outdir : this path is to the place where all the files will be saved during

```

1
2
3
4
5
6
7
8
9      &control
10         calculation='scf'
11         restart_mode='from_scratch',
12         pseudo_dir = '/path/to/the/potential/file/',
13         prefix='NANO'
14         tprnfor = .true.
15         max_seconds=82000
16         outdir = /current/work/directory
17     /
18     &system
19         ibrav=8
20         celldm(1)=40
21         celldm(2)= 0.115981
22         celldm(3)=1.00
23         nat=28
24         ntyp=1
25         ecutwfc=60.0
26         ecutrho=240
27         input_dft='vdw-df-c09'
28     /
29     &electrons
30         mixing_mode='local-TF'
31         mixing_beta = 0.8
32         conv_thr = 1.0d-7
33     /
34     &ions
35     /
36     &cell
37     /
38     ATOMIC_SPECIES
39     C 12.0107 06-C.GGA.fhi.UPF
40     K_POINTS automatic
41     1 8 1 0 0 0
42     ATOMIC_POSITIONS (angstrom)
43     C      4.533520557 -0.015659299 1.396325317
44     C      3.919311645 -0.015456942 2.674702484
45     C      1.735328192 -0.015034893 4.412842012
46     C      0.352415152 -0.015029285 4.727001312
47     C_     -2.370427231 -0.015079345 4.100075934

```

Figure A.16: Input file for pw.x

and after the computation.

350 Appendix A.1.2. *&system*

- **ibrav** : defines the type of lattice that is used. In this example, 8 is an orthorhombic unit cell with basis vectors : $v_1 = (a, 0, 0)$, $v_2 = (0, b, 0)$, $v_3 = (0, 0, c)$.
- **celldm** : this part defines the crystallographic lattice constants. The de-

355 fault units are Bohr and not Angstroms (1 Bohr = 0.529177249 Angstrom).
 In the orthorhombic case, $\text{celldm}(1) = a$, $\text{celldm}(2) = b/a$ and $\text{celldm}(3)$
 = c/a . Here an explanation about the values shown in the example (fig.
 A.16) is necessary. In this example the vector along the length of the nan-
 otube is given by $\text{celldm}(2)$. The two others are set to a big value in order
 360 to "insulate" the system from boundary conditions effects ($\text{celldm}(1) = 40$,
 means that $a = 40$ bohr, and $\text{celldm}(3) = 1$, means that $c = 40$ bohr). To
 define the value that one has to put for $\text{celldm}(2)$, the way is to compute
 the distance between two consecutive rings (perhaps using a visualization
 program), convert this value to Bohr (1 nm = 18.8971 Bohr) and divide
 365 it by $\text{celldm}(1)$, because the measured value will give 'b', but $\text{celldm}(2) =$
 b/a .

- nat : is the number of atoms in the simulation. It has to be equal to the
 number of lines under the "ATOMIC_POSITIONS" section.
- ntyp : defines the number of different types of atoms present in the simu-
 370 lation.
- ecutwfc : is the cutoff for the kinetic energy of the wavefunctions.
- ecutrho : is the cutoff for the kinetic energy of the charge density and
 potential, the default value (if not specified) is $4 \cdot \text{ecutwfc}$.
- input_dft : defines the exchange and correlation functionals employed in
 375 the calculation. Here we chose the vdW-DF non-local correlation func-
 tional [42, 43] with the C09 [44] exchange functional in order to account
 for London dispersion interactions (or van der Waals forces) within our
 calculations.

Appendix A.1.3. ATOMIC_SPECIES

380 Under this header one has to specify the type of atom, the mass of the
 atom and the exact name of the pseudopotential that will be used for this atom
 (located in 'pseudo_dir').

Appendix A.1.4. ATOMIC_POSITIONS

These are the coordinates of the atoms present in the system. There are many ways to do this; we just specified the type of atom (Carbon in the example Figure, A.16) and the coordinates using an .xyz format without the two first lines. After ATOMIC_POSITIONS, we specified the unit (see documentation for more details). For the structures here we employed Cartesian coordinates in angstroms.

Appendix A.2. Run pw.x

Now that the input file is created, one can run pw.x. Using MPI (on TAMNUN or a similar SYSTEM) the line to execute the code would be :

```
mpirun -np n /usr/local/espresso/bin/pw.x < name.in > name.out
```

where n has to be replaced by the number of processors. The path before pw.x is the right one for TAMNUN but could differ on another system. The result of this first part should be a folder with the name 'prefix'.save, where 'prefix' is the name chosen for the prefix, (NANO in our example) and a file name.out.

Appendix A.2.1. vdw-DF calculations

We have included dispersion (van der Waals) interactions within our DFT calculations using the van der Waals density function (vdW-DF) [42, 43, 44]. To include these interactions in QE it is necessary to first generate the vdW-DF kernel table. This can be done using the following procedure:

- copy the file from `usr/local/espresso/PW/src ./generate_vdW_kernel_table.x` into the user's directory.
- go into the destination folder and use the command (note the . at the end, it is important):

```
cp usr/local/espresso/PW/src/generate_vdW_kernel_table.x .
```

- run it with : `./generate_vdW_kernel_table.x`

- move the resulting table file (*vdW_kernel_table*) into the directory where the pseudopotentials are located.

Remark : it can take a moment to execute the *generate_vdW_kernel_table.x* file but it only needs to be generated once. If this is not done properly you may get the following error while running pw.x:

Error in routine read_kernel_table (1) :

No \ "vdW_kernel_table \ " file could be found

If you see this error try to repeat the above directions carefully.

Appendix A.3. Systems of multiple rings

Adding more rings is not totally trivial and should be omitted in a first calculation. As explained in section Appendix A.1.2, it is clear that if the number of rings changes, the number of atoms will change (*nat*) and the unit cell length along the b-axis, *celldm(2)* will also change. For example, if the system is now a 6-rings system the value of *celldm(2)* has to be multiplied by 6.

The position of the atoms is another aspect that requires care. Indeed, the *celldm* vectors define the simulation box. We recall the values are given in Bohr. On the other hand, the values of the positions are given in Angstrom. The fact that the positions are in Angstroms is a special case, it could be defined in Bohr if we wanted to. So an important thing to check is that all the atoms are contained in the volume defined by the *celldm* vectors. For simplicity our box starts from (0,0,0) and has the size of the '*celldm* vectors' values. (Remark : 1 Bohr = 0.5291 Angstrom). So if the atoms are not contained in these boundaries they have to be translated. To check, the y-coordinate of the atom (in Angstrom) has to be in the interval $[0, \text{celldm}(2) \cdot 0.5291 \cdot \text{celldm}(1)]$. N.B. The atoms just need to be within one unit cell of each other and could be defined starting at say $(-1/4, -1/4, -1/4)$.

The translation is done by a script that needs as input file a .xyz file with the coordinates. After downloading and opening the *many_rings* tar file,

- compile the code with : *g++ many_rings.cpp -o exec_many_rings*

- put the xyz file in the same folder and execute the code with : `./exec_many_rings`
- enter the exact name of the file (.xyz included)
- select to do a manual translation or automatic one

A file with the same name plus "_translated.xyz" will be created in the current directory. These are the new coordinates that will be used.

In the case one chooses to translate automatically, the program will find the minimum value in the first and last ring and translate every atom by this value along the y direction in order to have one atom at the origin and all the others contained inside of the simulation box.

Appendix A.4. System under strain

The third case that has to be considered with care is the case where there is a strain in the nanotube. This can also be omitted in a first trial. The consequence of the strain will be to change the distance between the atoms. When one adds a strain in the nanotube the distance along y is increased and the distance along x and z (direction of the radius) is decreased. As the constraint is constant the distance between two neighbor-cells along one given direction should be the same. Nevertheless, as one can probably notice in Figure 4 the slice of the nanotube does not fall along a perfectly straight line, which means that the distance is not constant everywhere between the consecutive cells. This problem is due to the way that the nanotube was built numerically and more specifically due to the boundary conditions. A way to minimise this problem is to take the atoms in the middle of the nanotube in order to be as far as possible from the edges. Every ring has 28 atoms. For the nanotube presented in Figure 4, if one wants to select a system with 3 rings, the way to select them in the middle is to remove the 18 first rings removing the 504 first coordinates, then jump 84 coordinates and remove the rest. One can check that the distance is now constant between all the atoms. This has been tested for a system of 3 rings under 2.5, 5 and 10 percent strain. For a larger number of rings this could be problematic if the distance is not constant.

Appendix A.5. Input file for pp.x

After the first step, one should have the two files mentioned above. The next step is to extract the charge density from the output file. To do so, one will create an input file for pp.x and obtain an output file that will contain the charge density for each point of the grid, as shown in our later figures. In detail:

```
&inputpp
  prefix = 'NANO'
  filplot = 'name.charge'
  plot_num = 0
  outdir = /current/work/directory
/
&plot
  nfile = 1
  filepp(1) = 'name.charge'
  weight(1) = 1.0
  iflag = 3
  output_format = 6
  fileout = 'name.charge.density.dat'
/
```

Figure A.17: Input file for pp.x

Appendix A.5.1. *&inputpp*

- prefix : as mentioned before it is really important that the name of the prefix is exactly the same as it was for pw.x.
- filplot : this is the name of the outputfile and can be changed.
- plot_num : defines the quantity that one is interested in during the post-processing, 0 is for the charge density (see the QE documentation for other values).
- outdir : same as before.

Appendix A.5.2. *&plot*

This section may not be needed for other systems but from our experience on TAMNUN it is better to specify it.

- `filepp(1)` : is the name of the output file that will contain the quantity to be plotted and saved in `fileout`, here it is the charge density for example.
- `output_format` : the integer defines the format of the output file (see documentation).
- `fileout` : name of the file that contains the data to do the plot. This is not used for AViz but it can be useful if one wants to use other software such as `xcrysden` [45].

Appendix A.5.3. Remark

On TAMNUN it is important to put the `outdir` at the end of the section `$inputpp`. Another observation is that it is possible to specify the amount of data saved during the computation using `'disk.io'`. If this is not specified the default value is 'low' but the less data saved the more RAM is used on some systems and the code can crash/terminate if the available RAM is not sufficient. The default value is advised.

Appendix A.6. Run `pp.x`

The execution line is similar to the previous one :

```
mpirun -np n /usr/local/espreso/bin/pp.x < name.in > name.out
```

Appendix A.7. Example of a 1 ring system

For the one ring system (illustrated in Figure 7), the tar file contains an input file for `pw.x` and another one for `pp.x` and also two bash scripts that create these files and submit them through the queuing system on TAMNUN.

Appendix A.8. Creation of the `.xyz` file for the charge density

The output file from `pp.x` should now be `name.charge`. This file has the structure shown in Figure A.18. The first line has 8 numbers, the three first numbers are the size of the grid in the three spatial directions, the three following are the same repeated. The seventh number corresponds to the number of atoms

and the last one to the number of different type of atoms. On the second line, the first number is the type of Bravais lattice, and the three following numbers are the celldm defined in the input file of pw.x. The first lines relate to the lattice constants. The charge density, is defined for each point of the grid starting at line 34.

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

```

1	200	24	200	200	24	200	28	1	0.00000000	0.00000000	0.00000000
2	8	40.00000000	0.11598100	1.00000000	0.00000000	60.0000000000	0	0	0.00000000	0.00000000	0.00000000
3	9726.8336296644	4.0000000000	4.0000000000	4.0000000000	4.0000000000	4.0000000000	4.0000000000	4.0000000000	4.0000000000	4.0000000000	4.0000000000
4	1	C	4.00								
5	6	1	0.214965823	0.000000000	0.066382059	1					
6	7	2	0.185958603	0.000000000	0.126744809	1					
7	8	3	0.082164899	0.000000000	0.209390854	1					
8	9	4	0.016862364	0.000000000	0.224250871	1					
9	10	5	-0.112524912	0.000000000	0.194802195	1					
10	11	6	-0.164934457	0.000000000	0.153071456	1					
11	12	7	-0.222431641	0.000000000	0.033484397	1					
12	13	8	-0.222431641	0.000000000	-0.033484397	1					
13	14	9	-0.164934457	0.000000000	-0.153071456	1					
14	15	10	-0.112524912	0.000000000	-0.194802195	1					
15	16	11	0.016862364	0.000000000	-0.224250871	1					
16	17	12	0.082164899	0.000000000	-0.209390854	1					
17	18	13	0.185958603	0.000000000	-0.126744809	1					
18	19	14	0.214965823	0.000000000	-0.066382059	1					
19	20	15	0.222431641	0.057990433	0.033484397	1					
20	21	16	0.164934457	0.057990433	0.153071456	1					
21	22	17	0.112524912	0.057990433	0.194802195	1					
22	23	18	-0.016862364	0.057990433	0.224250871	1					
23	24	19	-0.082164899	0.057990433	0.209390854	1					
24	25	20	-0.185958603	0.057990433	0.126744809	1					
25	26	21	-0.214965823	0.057990433	0.066382059	1					
26	27	22	-0.214965823	0.057990433	-0.066382059	1					
27	28	23	-0.185958603	0.057990433	-0.126744809	1					
28	29	24	-0.082164899	0.057990433	-0.209390854	1					
29	30	25	-0.016862364	0.057990433	-0.224250871	1					
30	31	26	0.112524912	0.057990433	-0.194802195	1					
31	32	27	0.164934457	0.057990433	-0.153071456	1					
32	33	28	0.222431641	0.057990433	-0.033484397	1					
33	34	3.79973317E-07	3.534424144E-07	4.060360547E-07	4.862195017E-07	6.102226259E-07					
34	35	8.057984030E-07	1.075203757E-06	1.426491118E-06	1.926447837E-06	2.685118826E-06					
35	36	3.788108248E-06	5.293774818E-06	7.342655696E-06	1.023622161E-05	1.431015908E-05					
36	37	1.983617681E-05	2.719797222E-05	3.719400913E-05	5.096073078E-05	6.971721061E-05					
37	38	9.498778844E-05	1.294406020E-04	1.773336299E-04	2.439870778E-04	3.355283334E-04					
38	39	4.602727457E-04	6.309783461E-04	8.652204943E-04	1.183620455E-03	1.608964787E-03					
39	40	2.168273005E-03	2.895263679E-03	3.828220999E-03	5.002528993E-03	6.441942626E-03					

Figure A.18: Output file from pp.x, 40 first lines.

Our aim is to create a .xyz format file with the charge density. More precisely, each value of charge density given in the output file corresponds to a point in the grid. To convert this file into an appropriate format one can download a script written in C++ as part of the example.input file and carry out the following:

1. compile and execute the code : `g++ charge_density.xyz.cpp -o exec_charge_density`
2. copy the file you wish to process into the directory where the script is
3. run `./exec_charge_density` to open the program
4. enter the name of the file to convert and press enter

A new file with the right format is now created in the same folder and is ready to be visualized. An older fortran conversion file for QE output is at [2].

Appendix B. Further data processing and visualization

Appendix B.1. Initial visualization with AViz

As it was mentioned previously, the raw charge density output from QE for this system is displayed in four different parts (see Figure B.19). This is a special case and could have been done differently.

Figure B.19: Charge density for a single ring of nanotube - initial visualization with AViz.

Appendix B.1.1. Input file for AViz

The output file after the execution of the C++ routine is a standard .xyz file for AViz (see Figure B.20 and Section 5) having the first line with an integer total number of dots, then a comment line and then the third and subsequent lines containing the information. All numbers from line 3 onward have to be real ones. The different columns in our example are :

1. A letter (a,b,c,d,e,f) that indicates the colors (transparent, black, red, green, yellow, blue)
2. The x coordinate
3. The y coordinate

- 540 4. The z coordinate
5. The charge density at that point

```

1 | 960000
2 | colored
3 | a 1. 1. 1. 3.37997332E-07
4 | a 2. 1. 1. 3.53442414E-07
5 | a 3. 1. 1. 4.06036055E-07
6 | a 4. 1. 1. 4.86219502E-07
7 | a 5. 1. 1. 6.10222626E-07

```

Figure B.20: Input .xyz file for AViz, containing a letter for color selection, coordinates and charge density.

In this file the coordinates go from 1 to 200 in each direction. The colors are in order of increasing density ranging from less than 0.0005 to 0.5 or above in units of a_0^{-3} , where a_0 is the Bohr radius, see Figure B.21.

Figure B.21: Color key on the left, one quarter of a nanotube with this color scheme on the right.

545 In Figure B.21 we show the color key and a realization of one quarter.

Appendix C. C++ code to recombine the circle

In this part the different important functions implemented will be presented. The general procedure is to read the file mentioned above, find which part corresponds to which quarter of circle, recombine them and create a new input file for AViz.

Appendix C.1. Function to read the initial .xyz file

The first step is to read the file in order to work on it after. To do so, it is possible to read line after line and store each line in a vector of string (Figure C.22).

```
void read_from_the_file(vector<string>& vector_file, string file_name)
{
    std::ifstream file(file_name);
    if(file.is_open())
    {
        string new_line;
        while (getline(file, new_line))
        {
            vector_file.push_back(new_line);
        }
    }
    else cout << "Unable to open file";
    file.close();
}
```

Figure C.22: Code to read from a file and create a vector of strings.

Appendix C.2. Separate the four parts

Once the vector of string is created, the second step is to separate the four different quarters. As mentioned above, the coordinates go from 1 to 200. The structure of the .xyz file is the following : taking half of the lines in the file ($\frac{total_number_lines}{2}$) corresponds to take half of the sample (cut in the z-direction). The result is a parallelepiped of dimension $(\frac{x}{2}, y, z)$. Indeed, if one takes the 200 first lines of the file, it corresponds to one single line on the plane (a one dimensional path). The 200 following lines represent the same line but translated in the y-direction. That means the second layer of points forming the plane.

This means that in order to separate the four parts, one has to take the lines labeled from 1 to 100 for the first half of the file (part1), the lines from 101 to 200 for the first half of the file (part2) and the same for the second part of the file (part3 and part4). This work is done by the loop shown in Figure C.23. Each part is stored in a vector of strings.

Appendix C.3. Recombine the circle

This is the most difficult part. The elements of the different vectors are strings but we have to extract the numbers for the coordinates and also the charge density, process the coordinates and recreate the file. To read the numbers from a string, one can use the function *sscanf()*, but the input line to read has to be a *char*. So the idea is to read each line of the vector, convert the string into a *const char* using *.c_str()*, extract the different variables in the appropriate format, modify the coordinates and store the line in the same format as it was initially. One has just to be carefull to erase the string and char variables after each iteration. (See Figure C.24.)

Appendix C.4. Dilute points

The sample is quite thick and the points are dense but in order to see the physics "inside" the sample one needs to dilute the points randomly. The idea is to delete a given number of points randomly in the vector that contains the information about the coordinates and charge density. (See Figure C.25).

Appendix C.5. Create the final output

The last step is to write an output file with the *.xyz* format described above but with the changed coordinates to have the right circle. This can be done using *ofstream*. The two first lines contain the number of points followed by a comment line. The rest can be done with a loop reading each element of the vector of strings recombined and writing it in the output file. (See Figure C.26).

```

1
2
3
4
5
6
7
8
9
10 for (unsigned int i(0); i<number_points/100 ; i=i+2)
11 {
12     // upper part
13     if (i < number_points/200)
14     {
15         // right part
16         incr = 100*i+1;
17         incr = incr + 1; // because of the two first lines
18         while (incr <= (i+1)*100+1)
19         {
20             vector_part1.push_back(vector_file[incr]);
21             incr = incr + 1;
22         }
23         // left part
24         incr = (i+1)*100+1;
25         incr = incr + 1; // because of the two first lines
26         while (incr <= (i+2)*100+1)
27         {
28             vector_part2.push_back(vector_file[incr]);
29             incr = incr + 1;
30         }
31     }
32
33     else if (i>= number_points/200)
34     {
35         // right part
36         incr = 100*i+1;
37         incr = incr + 1; // because of the two first lines
38         while (incr <= (i+1)*100+1)
39         {
40             vector_part3.push_back(vector_file[incr]);
41             incr = incr + 1;
42         }
43         // left part
44         incr = (i+1)*100+1;
45         incr = incr + 1; // because of the two first lines
46         while (incr <= (i+2)*100+1)
47         {
48             vector_part4.push_back(vector_file[incr]);
49             incr = incr + 1;
50         }
51     }
52 }
53
54
55
56
57
58
59
60
61
62
63
64
65

```

Figure C.23: Part of the code to separate the quarters of the sample.


```

1
2
3
4
5
6
7
8
9     char color[2];
10    double x(0),y(0),z(0);
11    float charge(0.0);
12    std::string str("");
13    std::string str0("");
14
15    const char *cstr("") ;
16    stringstream ss0;
17
18    for (unsigned int i(0); i<vector_part.size(); i++)
19    {
20
21        str = vector_part[i];
22        cstr = str.c_str();
23
24        sscanf(cstr, "%3s %lf %lf %lf %E",color,&x,&y,&z,&charge);
25        if (number_part==1)
26        {
27            ss0 << " " << color << " " << x << ". " << y << ". " << z << ". " << charge;
28        }
29        else if (number_part==2)
30        {
31            ss0 << " " << color << " " << x-200. << ". " << y << ". " << z << ". " << charge;
32        }
33        else if (number_part==3)
34        {
35            ss0 << " " << color << " " << x << ". " << y << ". " << z-200. << ". " << charge;
36        }
37        else if (number_part==4)
38        {
39            ss0 << " " << color << " " << x-200. << ". " << y << ". " << z-200. << ". " << charge;
40        }
41        str0 = ss0.str();
42
43        vector_ordered.push_back(str0);
44
45        ss0.str("");
46        str0.clear();
47
48    }
49

```

Figure C.24: Code to recombine the four parts of the sample and create a vector containing the information.

590 Appendix C.6. Run the code and results

To run the code, one has to compile it first using the command :

g++ -o nano4parts nano4parts.cpp.

```

vector_dilute = vector_ordered;
int element_to_delete(0);

for (unsigned int i(0); i<dilution_number; i++)
{
    element_to_delete = rand()%(vector_dilute.size()-2)+1;
    if (element_to_delete > vector_ordered.size())
    {
        cout << "size = " << vector_dilute.size() << " random number = " << element_to_delete << endl;
    }
    vector_dilute.erase(vector_dilute.begin() + element_to_delete);
}

```

Figure C.25: Code to dilute the points and see through the sample.

```

string output_name;
output_name = file_name + "_" + part_nb + ".xyz";
std::ofstream file_out;

file_out.open(output_name);

file_out << vector_part.size() << endl;
file_out << "colored part " + part_nb << endl;

for (unsigned int i(0); i<vector_part.size(); i++)
{
    file_out << vector_part[i] << endl;
}
file_out.close();

```

Figure C.26: Code to create the output file with the changed coordinates.

This will create an executable file named *nano4parts* which should be placed in the same folder as the initial input file executed. The name of the input file will be requested, enter it (include the extension in the name : example.xyz) and press enter. The code generates five .xyz files : four to visualize each of the four parts of the sample and the fifth is the full circle. They will be generated in the same folder. One can then visualize the .full.circle.xyz file with AViz and obtain the result presented in Figure 8.

An easy way to view the inside of the sample is to make a random dilution. After this random dilution, one obtains the result shown in Figure 9.

Appendix C.7. Download : Recombine circle

The C++ code is downloadable on [7].

In order to compile and execute the program, one has to :

1. type `g++ -o name_of_executable nano4parts.cpp`
2. be sure that the file with the data is in the same folder
3. run the program using : `./name_of_executable`

REFERENCES

- [1] J. Adler, "Visualization in Atomistic and Spin Simulations" Computers in Science and Engineering, (2003) **5** 61-65.
- [2] <http://phony1.technion.ac.il/~aviz>
- [3] J. Adler, A. Hashibon, N. Schreiber, A. Sorkin, S. Sorkin and G. Wagner, "Visualization of MD and MC Simulations for Atomistic Modeling" Computer Physics Communications, (2002) **147** 665-9.
- [4] S. Johnson, W. Potter, K. Malkin, Proceedings of the International Conference on Simulation in Engineering Education, (1994) **26** 199-202.
- [5] <http://phjoan23.technion.ac.il/~phr76ja/joeyfox/Hydrogen.html>
- [6] J. Adler, J. Fox, R. Kalish, T. Mutat, A. Sorkin and E. Warszawski "The essential role of visualization for modeling nanotubes and nanodiamond", Computer Physics Communications (2007) **177** 19-20.
- [7] <http://phony1.technion.ac.il/~edviz>
- [8] CPC program library.
- [9] <https://github.com/simphony/AViz>
- [10] <http://phelafel.technion.ac.il/~orcohen/DFTVisualize.html>
- [11] <http://classic.chem.msu.su/gran/gamess/index.html>

- [12] J. Adler, J. Zaffran, A. Silverman, A. Sorkin, O. Cohen and R. Kalish, "Simulation and visualization of nanodiamond and nanographite", Computer Physics Communications, (2011) **182** 2009.
- [13] <http://phycomp.technion.ac.il/~jeremie>
- [14] G. Kresse and J. Furthmüller, "Efficient iterative schemes for *ab initio* total-energy calculations", Phys. Rev. B. (1996) **54** 11 169.
- [15] G. Kresse and D. Joubert, "From ultrasoft potentials to the projector augmented-wave method", Phys. Rev. B. (1999) **59** 1758.
- [16] <http://jp-minerals.org/en/>
- [17] J. C. Sprott "Simple programs create 3D Images" Computers in Physics, (1992) **6** 132-8.
- [18] <http://phelafel.technion.ac.il/~peledan>
- [19] D. Peled, A. Silverman and J. Adler, "3D visualization for atomistic simulations on every desktop", IOP Conference Series, (2013) **454**, p. 012076.
- [20] J. Adler, Y. Koenka and A. Silverman, "Adventures in carbon visualization with AViz" *Physics Procedia* (2011) **15** pp 7-16.
- [21] <http://phelafel.technion.ac.il/~meytal>
- [22] Joan Adler, Yaron Artzi, Liron ben Bashat, Tzipora Yael Izraeli, Meital Kreif, Ido Lavi, Alexander Leibenzon, Adam Levi, Itai Schlesinger, Elad Toledano, Uria Peretz, Yonatan Weisler and Alon Yagil, "How do I simulate problem X?", IOP Conference Series, (2014) **510** 012003.
- [23] J. Adler and P. Pine, "Visualization Techniques for Modelling Carbon Allotropes", Computer Physics Communications, (2009) **180** 580-2.
- [24] P. Pine, Y. Yaish and J. Adler, "Simulation and vibrational analysis of thermal oscillations of single-walled carbon nanotubes", Phys. Rev. B (2011) **83** 155410.

- [25] P. Pine, Y. Yaish and J. Adler, “Thermal oscillations of structurally distinct single-walled carbon nanotubes”, Phys. Rev. B (2011)**84** 245409.
- [26] P. Pine, Y. Yaish and J. Adler, “The affect of boundary conditions on the vibrations of single-walled carbon nanotubes”, J. App. Phys., (2011), **110** 124311, Also featured in Virtual Journal of Nanoscale Science and Technology, **25** no 2.
- [27] P. Pine, Y. Yaish and J. Adler, “Simulation of nanosensors based on single walled carbon nanotubes”, IOP Conference Series, (2012) **402** 012002.
- [28] P. Pine, Y. Yaish and J. Adler, “Vibrational analysis of thermal oscillations of single-walled carbon nanotubes under axial strain”, Phys. Rev. B, (2014) **89** 115405.
- [29] P. Pine, Ph. D. thesis, Technion, 2012, <http://phycomp.technion.ac.il/~newphr76ja/polina.pdf>
- [30] B. I. Yakobson, C.J. Brabee and J. Bernholc. “Nanomechanics of Carbon Tubes:Instabilities beyond Linear Response”, Phys. Rev. Lett. (1996) **76** 2511.
- [31] Y. Huang, J. Wu and K.C. Hwang, “Thickness of graphene and single-wall carbon nanotubes”, (2006) Phys. Rev. B **74**, 245413.
- [32] <http://www.simphony-project.eu>
- [33] D. W. Brenner, “Emprical potential for hydrocarons for use in simulating the chemical vapor deposition of diamond films”, Phys. Rev. B (1990) **42** 9458.
- [34] http://phycomp.technion.ac.il/~tamnun/easy_lammps.html
- [35] Joan Adler, Hila Glanz, and Nadir Izrael, “A “Rosetta stone” for AViz”, Physics Procedia, (2014), **57c** 2-6. (DOI10.1016/j.phpro.2014.08.122)
- [36] <http://phycomp.technion.ac.il/~tamnun/qe/draw.html>

- [37] <http://www.quantum-espresso.org>
- [38] P. Giannozzi et al. “QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials”, J. Phys. C (2009) **21**, 395502.
- [39] http://www.quantum-espresso.org/wp-content/uploads/Doc/INPUT_PW.html#idm1680
- [40] http://www.quantum-espresso.org/wp-content/uploads/Doc/INPUT_PP.html
- [41] <http://phony1.technion.ac.il/~edviz/bastien2.gif>
- [42] M. Dion, H. Rydberg, E. Schröder, D. C. Langreth, and B. I. Lundqvist, “Van der Waals Density Functional for General Geometries” Phys. Rev. Lett (2004) **92** 246401.
- [43] T. Thonhauser, V. R. Cooper, Shen Li, Aaron Puzder Per Hyldgaard and D. C. Langreth, “Van der Waals density functional: Self-consistent potential and the nature of the van der Waals bond”, Phys. Rev. B, (2007) **76** 125112.
- [44] V. R. Cooper, “Van der Waals density functional: an appropriate exchange functional”, Phy. Rev. B (2010) **81** 161104.
- [45] <http://www.xcrysden.org/XCrySDen.html>