| Code Generalities | | | | |
|---|---|---|---|---|
| Code Name | Purpose | License | Web Site | Dictator Code Name |
| Abinit | ABINIT is a package whose main program allows one to find the total energy, charge density and electronic structure of systems made of electrons and nuclei (molecules and periodic solids) within Density Functional Theory (DFT), using pseudopotentials and a planewave basis. ABINIT also includes options to optimize the geometry according to the DFT forces and stresses, or to perform molecular dynamics simulations using these forces, or to generate dynamical matrices, Born effective charges, and dielectric tensors. Excited states can be computed within the Time-Dependent Density Functional Theory (for molecules), or within Many-Body Perturbation Theory (the GW approximation). In addition to the main ABINIT code, different utility programs are provided. | GNU-GPL | http://www.abinit.org | Xavier Gonze |
| DP | DP is a Linear Response TDDFT code, in Frequency-Reciprocal space, on a Plane Waves basis set. It allows to calculate Dielectric and Optical Properties (Optical Absorption, Reflectivity, Refraction Indices, EELS, IXSS, CIXS, ...). Systemms taken into account are: Bulk Solids, Surfaces, Clusters, Molecules, Atoms (through supercells) made of Insulator, Semiconductor and Metal elements. Approximations used: RPA, ALDA, GW-RPA, LRC, non-local kernels, ..., with and without LF (Local Fields). | SSOSAF (Open Source, Free and explicitly Gratis for Academic Purposes. Does not comprise Military and Commercial purposes). | http://www.dp-code.org | Valerio Olevano |
| EXC | EXC is a code implementing the Bethe-Salpeter equation method and working in reciprocal space, frequency domain, and using plane waves basis. Its purpose is to calculate, ab initio, the dielectric and optical properties, like absorption, reflectivity, refraction index, electron and X-ray energy loss, for a large variety of systems, ranging from bulk systems, surfaces, to clusters or atoms (using the supercell method), by solving the Bethe-Salpeter equation. The approximations used are the Tamm-Dankov or full coupling, switch off of the e-h interaction, switch off of the local-field effects. | GNU-GPL | http://www.bethe-salpeter.org | Francesco Sottile |
| FHI98PP | fhi98PP is a code to (i) generate norm-conserving pseudopotentials of the Hamann or Troullier-Martins type, and to (ii) check the transferability of the pseudopotential. | presently freely downloadable | http://www.fhi-berlin.mpg.de/th/fhi98md/fhi98PP | Martin Fuchs |
| Octopus | Octopus is a scientific program aimed at the ab initio virtual experimentation on a hopefully ever increasing range of systems types. Electrons are described quantum-mechanically within the Density-Functional Theory (DFT), in its time-dependent form (TDDFT) when doing simulations in time. Nuclei are described classically as point particles. Electron-nucleus interaction is described within the Pseudopotential approximation. | GNU-GPL | http://www.tddft.org/programs/octopus | Miguel Marques |
| V_Sim | V_Sim visualizes atomic structures such as crystals, grain boundaries and so on. The rendering is done in pseudo-3D with colored sphere to represent the atoms. The user can interact through many functions to choose the view, the size of the atoms, their color, the background color, the type of fog... Moreover V_Sim allows to export the view as image in GIF, PS or more formats. It has several solid state capabilities such as planes or iso-densities. | CeCILL | http://www-drfmc.cea.fr/sp2m/L_Sim/V_Sim/index.en.html | Damien Caliste |
| pspconvert | pspconvert.py is a command-line program to convert pseudopotential files. | GNU-GPL | http://www.etsf.eu/index.php?page=tools | Micael Oliveira |
| TOSCA | Tosca contains two different programs: a part of a car-parinello code CP for groundstate calculation and the calculation of the transition matrixelements in independent-particle approximation with and without realspace cutoff and a small postprocessing code which reads the matrixelements and which calculates various kinds of spectra ( Im(epsilon), Re(epsilon), RAS, and so on ). | GNU-GPL | http://www.users.unimi.it/etsf/tosca.html for the CP code | Giovanni Onida for CP, Katalin Gaal-Nagy for the postprocessing tool |
| EXCITING | EXCITING is an all-electron FP-LAPW code with many features including non-collinear magnetism, spin-spirals, Hartree-Fock, optimised effective potential (OEP), reduced density matrix functional theory (RDMFT), forces, structural optimisation, LDA+U, phonon dispersions, electron-phonon coupling, ELNES, linear optics and MOKE. Most features work together in combination. | GNU-GPL | http://exciting.sourceforge.net/ | Kay Dewhurst |
| Yambo | Yambo is a multipurpose Ab-Initio Many-Body code. A part of the Yambo code is released under the GPL license and can be downloaded from the web-site or from the SVN repository. This GPL version can calculate (i) Quasiparticles within the plasmon-pole GW and the COHSEX approximation, (ii) Response functions: Random-Phase approximation, Bethe-Salpeter approach and TDLDA. It also contains a post-processor that can be used to plot excitonic and electronic wavefunctions as well as other quantities. The Yambo code as a whole adds several features to the GPL version: (iii) electronic lifetimes within the GW approximation, (iv) Quasiparticles calculated on the real axis, (v) Response functions using the Nanoquanta TDDFT kernel, (vi) Total Energy: Adiabatic connection fluctuation-dissipation formula, (v) Self-consitent Exact Exchange, Hartree-Fock and Generalized Kohn-Sham (COHSEX), (vi) electron-phonon coupling (energies and lifetimes), (vii) Real-Time propagation. In addition Yambo includes spin (collinear and non-collinear) support and it can treat in a self-consistent manner the interaction with external Magnetic fields. Yambo can work at finite temperatures. Yambo takes as input the non-interacting electronic states calculated with other freely available ab-initio codes. The current version is capable of importing data from Abinit, PWscf and Kohn-Sham data in the ETSF netCDF format. | The public version of Yambo is released under the GNU-GPL | http://www.yambo-code.org | Andrea Marini |
| ELK | Elk is an all-electron full-potential linearised augmented-plane wave (FP-LAPW) code with many advanced features. The code is designed to be as simple as possible so that new developments in the field of density functional theory (DFT) can be added quickly and reliably. | GNU-GPL | http://elk.sourceforge.net | Kay Dewhurst |
| APE | APE (Atomic Pseudopotential Engine) is a tool for generating atomic pseudopotentials within the Density-Functional Theory framework. The program can create pseudopotential files suitable for the most widely used ab-initio packages, and, besides the standard non-relativistic Hamann and Troullier–Martins potentials, it can generate pseudopotentials using the relativistic and semi-core extensions to the Troullier–Martins scheme. | GNU-GPL | http://www.tddft.org/programs/APE/ | Micael Oliveira |

| Code Name | Documentation | | | |
|---|---|---|---|---|
| | Choice of the names | Comments | Internal description | Users documentation |
| Abinit | (Might be better ...) Usually input variable and subroutine names are between 6 and 9 characters long. So there are many abbreviations that might not be self-descriptive enough. For structured datatypes, the names are longer and more self-descriptive. For internal names, we forbid explicitely 1-letter name. Convention for variables related to dimensions and do-loop : suppose we consider the number of bands, then the maximal number of bands is mbands, the actual number of bands (for the k point) is nband, and the loop index is iband. | Usually, 1 or 2 comment lines every 10-20 lines of code. | Every routine has a header with usually 3-10 lines to explain the purpose of the routine, then a one-line description for each input and output argument. | Each input variable is described in a html document, available on the Web. Also on the Web, a beginners guide, a users guide, the html version of the source code (can be browsed). Other documents (.tex, .txt) are available in the package. |
| DP | The practice which is tried to be followed is that one standard in C, contracted names but still recognizable (ex. strcpy, strcat). Although this best practice is far to be applied everywhere. | The practice which is tried to be followed is again that one standard in C: a well written code self comments itself, and it does not need extensive commenting, apart from a comment line from time to time. Since this best practice relies on the "well-writing", which unfortunately is not always the case for this code, sometime the commenting is necessary. Anyway, the actual mathematical formula implemented are reported in comments in a more human form. | Again, also for this point we try to follow the best C practice, which tells self-explanating names of subroutines/functions and arguments, in the order expected by a C-minded programmer and self-explanating of their input or output character. This at least in non-specialized and very reusable routines. Those which do not respect the standard, of course need heavy description. | The user documentation is described in a very light html page online on the web. On the Web also a pdf document introducing more in details TDDFT and the actual DP implementation and use. Online a PDF document with some tutorials together with the files which go with. |
| EXC | The practice which is tried to be followed is that one standard in C, contracted names but still recognizable (ex. strcpy, strcat). Although this best practice is far to be applied everywhere. | The practice which is tried to be followed is again that one standard in C: a well written code self comments itself, and it does not need extensive commenting, apart from a comment line from time to time. Since this best practice relies on the "well-writing", which unfortunately is not always the case for this code, sometime the commenting is necessary. Anyway, the actual mathematical formula implemented are reported in comments in a more human form. | Again, also for this point we try to follow the best C practice, which tells self-explanating names of subroutines/functions and arguments, in the order expected by a C-minded programmer and self-explanating of their input or output character. This at least in non-specialized and very reusable routines. Those which do not respect the standard, of course need heavy description. | The user documentation is described in a very light html page online on the web. On the Web also a pdf document introducing more in details the Bethe-Salpeter method and the actual EXC implementation and use. Online a PDF document with some tutorials together with the files which go with. |
| FHI98PP | Names in main routines are meant to be descriptive but specific to fhi98pp, reflect coded formulas and symbols of pseudopotential theory. | Overall many explanatory comments, detail varies within the code. | See comments. | A published paper and a tutorial booklet describe the functionality/use of the code. |
| Octopus | Names are long and descriptive, both for subroutine names, types, input variables, etc. Subroutines typically start by a prefix that indicate the module where they are defined. Module names have the suffix '_m', whyle structures have the suffix '_t'. | Varies a lot in the code, being some parts better commented than others. | Again, we do not have strict rules concerning this. However, as names are usually quite descriptive, the need for this "description" is somehow lower. | We have a wiki that contains, among others, a user's manual and tutorials (being updated now). Variables are completely documented in the code using a simple language, and scripts are used to extract this information and show it in diverse formats, including the web. |
| V_Sim | Names are long and descriptive. It is based on the scheme partAction_details() where 'part' is the name of the part of the code (e.g. visuData), 'Action' the action of the routine (e.g. Get) and 'details' further explanation for the action (e.g. openGlView which will gives visuDataGet_openGlView). Structures are named with a leading capital letter. Core routines are all beginning with 'visu'. | Usually not as much as required. | The public routines are all documented using Gtk-doc in the header file. This documentation includes a description of the routine, of the input and output variables and the return value if any. | The user manual is made of a static HTML page available on the web site. |
| pspconvert | Names are usually long and descriptive. | There are very few comments in the code. | Most of the classes have a descriptive header. | There is a user's manual written in latex. |
| TOSCA | For CP: ask Giovanni, and for the postprocessing tool: names are choosen according to the names in reference articles, e.g., nv=number of valence bands | For the CP: ask Giovanni, for the postprocessing tool: usually there is an explanation at the begin of each subroutine, which are mostly short | see comments | For CP: manual on web page, for the postprocessing tool: there is a manual written in latex given as pdf document with the source |
| EXCITING | Short as possible while still conveying meaning. | Most parts of code have comments - probably not enough though. | Subroutines have a full LaTeX description using the ProTex system (http://gmao.gsfc.nasa.gov/software/protex/). Not all subroutines are documented yet. | User documentation incorporated in the code. Also available is a Wiki and FAQ. |
| Yambo | Names are normally long and descriptive. Routines, modules and variables with a common purpose are named using the same extension (like "IO" for all I/O related or "Dipole" for all linked to the calculations of dipoles and so on). | Varies a lot within the code, but overall comments within code are sparse. | Two PDF documents distributed with the code explains the internal structure of the code and the relationship between subroutines, modules and data storage. | User documentation online is quite extensive. Much of the physical basis behind the different code tasks is explained in LaTeX-formatted web pages composed by the developers and main users. Code tasks and variables in input files are given specific documentation, although this could be improved. A set of tutorials is also available, as used in the Benasque TDDFT school. An online user forum moderated by the developer team offers further dedicated help. Finally, a published paper offers a comprehensive overview of the theories implemented in the code and outlines the runtime procedure. |
| ELK | Names are short but convey meaning | most parts of the code have comments | routines have a header that explains their purpose and describes each input and output argument | The Elk Code Manual is available, as is a manual for the Spacegroup utility. The Elk forums on SourceForge.net or Wiki pages answer most questions. A mailing list is available for notification of new releases, features and bugs in the code. |
| APE | Names are short and somewhat descriptive | quite many comments and descriptions | routines have a header that explains their purpose and describes the arguments | Online documentation and PDF file, both quite extensive, describing installation, input options and several examples |

| Code Name | Standards | | | | | |
|---|---|---|---|---|---|---|
| | Compliance with F90 standards | Compliance with F95 standards | Compliance with ANSI C | Compliance with C++ standards | Strategy for Modularity | Version management software |
| Abinit | Yes | N/A | N/A | N/A | External libraries can be readily incorporated thanks to the autotools. Groups of ABINIT routines are gathered in thematic directories (not yet independent libraries, though). Usually, the printing tasks are separated from the compute task, but this is not always the case. The writing of big files, however, is completely separated from the computation, and one uses (among others the ETSF-IO library. | Bazaar |
| DP | Yes | N/A | Yes | N/A | The code is adaptive to a large range of external libraries, free or proprietary. Libraries are linked through wrapper modules, and the main part of the code does not specialize for a given library. The actual library to be linked and the wrapper to be compiled and linked are chosen at config time. | subversion |
| EXC | Yes | N/A | Yes | N/A | The code is adaptive to a large range of external libraries, free or proprietary. Libraries are linked through wrapper modules, and the main part of the code does not specialize for a given library. The actual library to be linked and the wrapper to be compiled and linked are chosen at config time. | subversion |
| FHI98PP | No | No | n/a | n/a | n/a | none |
| Octopus | Yes | We avoid using F95 due to some bad experience with some compilers | Yes | N/A | We make extensive use of external libraries and Fortran modules. The code is organized in a object oriented manner. | subversion |
| V_Sim | N/A | N/A | Yes | Yes (for plugins using C++) | There is a possibility of plugins to extend the capabilities, especially for actions using an external library like OpenBabel. | subversion |
| pspconvert | N/A | N/A | N/A | N/A | N/A | subversion |
| TOSCA | For CP: I don't think so, the postprocessing tool is written in f77 | | | | For CP: ask Giovanni, for the postprocessing tool: no libraries are necessary, however the code is modular and can be easily extended to other spectra based on matrixelements | nothing ... |
| EXCITING | Yes | N/A | N/A | N/A | Code uses BLAS/LAPACK, but is otherwise self-contained. Subroutines are divided into main and library types, the latter are independent of the rest of the code. | None |
| Yambo | F90 compliant except for use of "forall" statements. | OK | N/A | N/A | Groups of Yambo routines are gathered in thematic directories and joint together in independent libraries. Specific modules and interfaces take care of the I/O, comments and parallelization. | subversion |
| ELK | Yes | N/A | N/A | N/A | Code is self-contained but can use enhanced BLAS or FFT libraries | subversion |
| APE | N/A | Yes | Yes | N/A | N/A | subversion |

| Code Name | Libraries | | | | |
|---|---|---|---|---|---|
| | BLAS/LAPACK utilisation | NETCDF utilisation | ETSF-IO library | ETSF-XC library | Other |
| Abinit | Yes | Yes | | | FoX, Wannier90, ETSF-IO |
| DP | Yes | Yes | Yes but not yet the main ETSF-IO | No | MFFT, FFTW, CXML, ESSL, IMSL, ASL, NAG Goedecker-FFT |
| EXC | Yes | Yes | Not yet | No | MFFT, FFTW, CXML, ESSL, IMSL, ASL, NAG Goedecker-FFT |
| FHI98PP | no | no | no | no | n/a |
| Octopus | Yes | Yes | No | | fftw, trlan, arpack, sparskit, expokit, qshep, metis, poisson_isf. Furthermore, we have internal libraries (but independent) libstring (to handle passing strings from C to Fortran), liboct_parser (to parse the input file), and libxc (with xc functionals) |
| V_Sim | N/A | Yes (using a plugin) | No (waiting for C backend of ETSF_IO) | N/A | OpenBabel 2.x (plugin), OpenGL |
| pspconvert | N/A | N/A | N/A | N/A | GSL |
| TOSCA | For the CP: yes | no | not till now | For CP: no, for postprocessing: not necessary | |
| EXCITING | Yes | No | No | No | Optional use of FFTW |
| Yambo | Yes | Yes | Yes, a convertor is available, although not completely tested | Interface is ready, Yambo already uses the same definitions included in the xc library for XC functionals. Still the full implementation must be finalized. | FFTW, goedecker-FFT, SCALAPACK, BLAS, LAPACK, netCDF, BLACS, MPI |
| ELK | Yes | No | No | Yes as an option | FFTPACK5 |
| APE | No | No | No | Yes | GSL |