Mirza Akbar Ali
Quantum Solid State Physics
QuanTEEM - M1 - S1

Computation of 3D Band Structure and Density of States (DOS) of 14 Face Centered Cubic (FCC) Crystals using Pseudopotentials

I. Introduction

ELECTRONIC properties of materials are crucial to their ability to function in a wide range of applications, from electronics and energy production to structural materials and biomedicine [1]. In order to understand and predict these properties, it is essential to study the energy levels and electron distribution within the material, which can be done through the study of the band structure and density of states (DOS) [2].

The band structure of a material describes the allowed energy levels that electrons can occupy within the material. These energy levels are formed by the interaction of the electrons with the crystal lattice and other electrons in the material. They are typically organized into bands separated by energy gaps. The band structure can be calculated using various theoretical methods, such as density functional theory (DFT) and the tight-binding model, and it can also be measured experimentally using techniques such as angle-resolved photoemission spectroscopy (ARPES) [2].

The density of states (DOS) is a measure of the number of electronic states available at a given energy level. It is typically calculated by considering the electronic states within a particular energy range, such as a band or an energy gap. The DOS can be used to calculate a range of electronic properties, such as the electrical conductivity and thermoelectric power, as well as to understand the electronic and optical responses of the material [3].

Both the band structure and DOS are important for understanding the electronic properties of materials and predicting their behavior in various applications. For example, the band structure can be used to predict the electrical conductivity of a material, as well as its suitability for use in electronics and optoelectronics. The DOS can be used to understand the optical properties of a material, such as its absorption coefficient and refractive index, as well as to predict its thermoelectric properties.

In addition to their theoretical and predictive capabilities, the study of the band structure and DOS also has important practical applications. For example, the band structure can design new materials with specific electronic properties, such as high electrical conductivity or optoelectronic performance. The DOS can be used to optimize materials for energy conversion and storage applications, such as thermoelectrics and batteries [4].

Overall, the study of the band structure and DOS is an essential part of materials science and engineering, with wide-ranging implications for various fields and applications. In this article, we will compute the three-dimensional electronic band structure and density of states (DOS) of 14 face-centred cubic (FCC) crystals using pseudopotentials.

II. Methodology

We adopt Bloch Model for the calculations of the band structures. The Bloch Model describes the spectrum of the electron energy states in the framework of the one-electron approximation using a periodic potential that is independent of time.

A. Bloch Model

Bloch's theorem applies to wave functions of electrons inside a crystal and rests in the fact that the Coulomb potential in a crystalline solid is periodic. As a consequence, the potential energy function, $V(\mathbf{r})$, in Schrödinger's equation should be of the form:

$$V(\mathbf{r}) = V(\mathbf{r} + \mathbf{R}_n) \tag{1}$$

Here, n points to a triplet of integer numbers (n_1, n_2, n_3) identifying a vector of the direct lattice $\mathbf{R}_n$, which is itself expanded on the direct lattice basis vectors $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$ according to:

$$\mathbf{R}_n = n_1\mathbf{a}_1 + n_2\mathbf{a}_2 + n_3\mathbf{a}_3 \tag{2}$$

The periodic $V(\mathbf{r})$ given in equation 1 may be expanded by a FOURIER series using the reciprocal lattice vector $\mathbf{G}$:

$$V(\mathbf{r}) = \sum_{\mathbf{G}} V_{\mathbf{G}} e^{i\mathbf{G}\cdot\mathbf{r}} \tag{3}$$

with the form factor of FCC lattice defined by

$$V_{\mathbf{G}} = V^S_{|\mathbf{G}|^2} \cos(\mathbf{G}\cdot\tau) + iV^S_{|\mathbf{G}|^2} \cos(\mathbf{G}\cdot\tau) \tag{4}$$

Where, with lattice spacing a, $\tau = \frac{a}{8}(1,1,1)$ and $-\tau$ are the positions of the two atoms relative to the centre of the primitive cell that is chosen as the origin.

The values of $V^S_{|\mathbf{G}|^2}$ and $V^A_{|\mathbf{G}|^2}$ are deduced from fit to experimental data. They feature $V_{\mathbf{G}} = 0$ if $|\mathbf{G}|^2 > 11(4\pi^2/a^2$ units).For a selection of 14 semiconductor, Table I shows the non-zero values for $|\mathbf{G}|^2 \leq 11(4\pi^2/a^2$ units). Even if the Fourier expansion of the potential is limited to $|\mathbf{G}|^2 \leq 11(4\pi^2/a^2$ units), a satisfactory converges requires that the representation of the Schrödinger equation in reciprocal space involves all $\mathbf{G}$ vectors such that $|\mathbf{G}|^2 \leq 21(4\pi^2/a^2$ units).

The potential $V(\mathbf{r})$ discussed above, being independent of time, allows for the separation of variables of the Schrödinger equation. Thus, finding the eigenenergies consists in solving the time-independent Schrödinger equation for wave function $\psi(\mathbf{r})$ expanded on the basis of Sommerfield free electron model given by:

$$\psi(\mathbf{r}) = \frac{1}{\mathcal{V}} \sum_{\mathbf{k}} c_{\mathbf{k}} e^{i\mathbf{k}\cdot\mathbf{r}} \tag{5}$$

Table I
LATTICE CONSTANT a IN ANGSTROMS AND PSEUDOPOTENTIAL FORM FACTORS, IN RYDBERG, DERIVED FROM THE EXPERIMENTAL ENERGY BAND SPLITTINGS [5].

	a	V_3^S	V_8^S	V_{11}^S	V_3^A	V_4^A	V_{11}^A
Si	5.43	-0.21	0.04	0.08	0.00	0.00	0.00
Ge	5.66	-0.23	0.01	0.06	0.00	0.00	0.00
Sn	6.49	-0.20	0.00	0.04	0.00	0.00	0.00
GaP	5.44	-0.22	0.03	0.07	0.12	0.07	0.02
GaAs	5.64	-0.23	0.01	0.06	0.07	0.05	0.01
AlSb	6.13	-0.21	0.02	0.06	0.06	0.04	0.02
InP	5.86	-0.23	0.01	0.06	0.07	0.05	0.01
GaSb	6.12	-0.22	0.00	0.05	0.06	0.05	0.01
InAs	6.04	-0.22	0.00	0.05	0.08	0.05	0.03
InSb	6.48	-0.20	0.00	0.04	0.06	0.05	0.01
ZnS	5.41	-0.22	0.03	0.07	0.24	0.14	0.04
ZnSe	5.65	-0.23	0.01	0.06	0.18	0.12	0.03
ZnTe	6.07	-0.22	0.00	0.05	0.13	0.10	0.01
CdTe	6.41	-0.20	0.00	0.04	0.15	0.09	0.04

We find the expression of Schrödinger equation in reciprocal space:

$$\left(\frac{\hbar^2 q^2}{2m_e} - E\right) c_{\mathbf{q}} + \sum_{\mathbf{G}} V_{\mathbf{G}} c_{\mathbf{q}-\mathbf{G}} = 0 \tag{6}$$

The system of equations given in equation 6 couples the values $\mathbf{q}$ differing from each other by a reciprocal lattice vector. Which, for $\mathbf{q} = \mathbf{k} - \mathbf{G}'$ and $\mathbf{G}' + \mathbf{G} = \mathbf{G}''$ can be written as

$$[\bar{H}(\mathbf{k}) - \mathbb{I}E]\bar{C}(\mathbf{k}) = 0 \tag{7}$$

with $\bar{C}(\mathbf{k}) = 0$, and $\bar{H}(\mathbf{k})$ is the Hamiltonian matrix in reciprocal space given by:

$$\begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots \\ \cdots & T_{\mathbf{k}+\mathbf{G}_2} & V_{\mathbf{G}_1} & V_{\mathbf{G}_2} & V_{\mathbf{G}_3} & V_{\mathbf{G}_4} & \cdots \\ \cdots & V_{\mathbf{G}_{-1}} & T_{\mathbf{k}+\mathbf{G}_1} & V_{\mathbf{G}_1} & V_{\mathbf{G}_2} & V_{\mathbf{G}_3} & \cdots \\ \cdots & V_{\mathbf{G}_{-2}} & V_{\mathbf{G}_{-1}} & T_{\mathbf{k}-0} & V_{\mathbf{G}_1} & V_{\mathbf{G}_2} & \cdots \\ \cdots & V_{\mathbf{G}_{-3}} & V_{\mathbf{G}_{-2}} & V_{\mathbf{G}_{-1}} & T_{\mathbf{k}-\mathbf{G}_1} & V_{\mathbf{G}_1} & \cdots \\ \cdots & V_{\mathbf{G}_{-4}} & V_{\mathbf{G}_{-3}} & V_{\mathbf{G}_{-2}} & V_{\mathbf{G}_{-2}} & T_{\mathbf{k}-\mathbf{G}_1} & \cdots \\ \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{8}$$

with

$$T_{\mathbf{k}-\mathbf{G}'} = \frac{\hbar^2 |\mathbf{k} - \mathbf{G}'|^2}{2m_e} + V_{\mathbf{G}=0}$$

Since $V((r)$ are real, it allows us to deduce $V_{-\mathbf{G}} = V_{-\mathbf{G}}^*$, which implies that the hamiltonian matrix is Hermitian. This coupled Hamiltonian matrix can be decoupled by setting $V_{\mathbf{G}=0} = V_0 = 0$, which corresponds to the constant adjusting the zero of the potential [6]. Thus plotting the band structure involves finding and plotting the eigenvalues of the decoupled Hamiltonian matrix.

B. Program Implementation for Band Structure Calculations

Complete program for the calculation of the band structures can be divided into the following steps.

- Step 1: Define the number of bands to be plotted, cutoff and maximum value of $\mathbf{G}$ described in II-A, lattice spacing and Pseudopotential Form Factors provided in Table I, positions of atoms (τ and $-\tau$) in the primitive cell.
- Step 2: Generate FCC lattice unit vectors in cartesian coordinates and volume of the primitive cell.
- Step 3: Generate FCC reciprocal lattice unit vectors in cartesian coordinates. Calculate the minimum norm of the reciprocal lattice unit vectors and defines the number of positive steps along each reciprocal lattice unit vector. Define the number of the reciprocal lattice vectors to be generated.
- Step 4: Generating reciprocal lattice vectors for calculations in all directions, sorting reciprocal lattice vectors by growing norm and keeping the reciprocal lattice vectors less than the cutoff limit.
- Step 5: Generate Brillouin Zone (BZ) exploration path according to traditional solid-state representation.
- Step 6: For the kept reciprocal lattice vectors, calculate the value of $V_{\mathbf{G}}$ defined in Eq 4.
- Step 7: Initialize Hamiltonian matrix and assign potential energy values corresponding to $\mathbf{G}$.
- Step 8: Calculate the difference $|\mathbf{k} - \mathbf{G}|^2$ along the BZ exploration path and kinetic energy part of the hamiltonian matrix, then diagonalize to find the eigenenergies.
- Step 9: Plot the eigenenergies against the BZ exploration path to obtain band structure.

C. Density of States (DOS)

Density of states refers to the number of states available per unit of energy, thus having units $[eV^{-1}]$. Unlike band structure calculations, DOS calculation is not restricted to exploring just the high symmetry path in BZ. We define a sequence of numbers

$$u_r = (2r - q - 1)/2q \qquad (r = 1, 2, 3, \cdots, q) \tag{9}$$

Where q is an integer that determines the number of special points in the set. With the above u_r's we define

$$\mathbf{k}_{prs} = u_p \mathbf{b}_1 + u_r \mathbf{b}_2 + u_s \mathbf{b}_3 \tag{10}$$

That gives q^3 distinct points in reciprocal space uniformly distributed in BZ [7]. Then, points in rest of the octants are generated by symmetry. We follow the same steps as outlined in II-B, except for the Step 5. Instead of exploring the BZ path along high symmetry points, we now perform the calculations for the points generated by Eq 10. Once eigenenergies have been calculated for all the points in BZ, we perform calculations for the DOS, which is to be understood as a distribution [6], given by

$$\mathcal{D}(E) = \sum_{\mathbf{k}} \delta(E - E_{\mathbf{k}}) \tag{11}$$

where

$$\delta(E - E_{\mathbf{k}}) = \frac{e^{-(E-E_{\mathbf{k}})^2/\sigma^2}}{\sigma\sqrt{\pi}} \tag{12}$$

III. RESULTS

Band structure and density of states (DOS) plots for energy range $-14\,eV$ to $6\,eV$ for 14 face centered cubic (FCC) crystals produced with the developed program are shown in Figures 1-28.

REFERENCES

[1] A.S.M.A. Haseeb. Electronic materials. In *Reference Module in Materials Science and Materials Engineering*. Elsevier, 2016.

[2] Neil W Ashcroft and N David Mermin. Solid state physics, cornell university, 1976.

[3] Richard M. Martin. *Electronic Structure: Basic Theory and Practical Methods*. Cambridge University Press, 2004.

[4] M. Jaros. *Concepts and Applications of Band Structure Engineering in Optoelectronics*, pages 147–163. Springer US, Boston, MA, 1991.

[5] Marvin L. Cohen and T. K. Bergstresser. Band structures and pseudopotential form factors for fourteen semiconductors of the diamond and zinc-blende structures. *Phys. Rev.*, 141:789–796, Jan 1966.

[6] Alain Dereux. *Selected Chapters of Solid State Physics*. Université de Bourgogne, 2022.

[7] Hendrik J. Monkhorst and James D. Pack. Special points for brillouin-zone integrations. *Phys. Rev. B*, 13:5188–5192, Jun 1976.

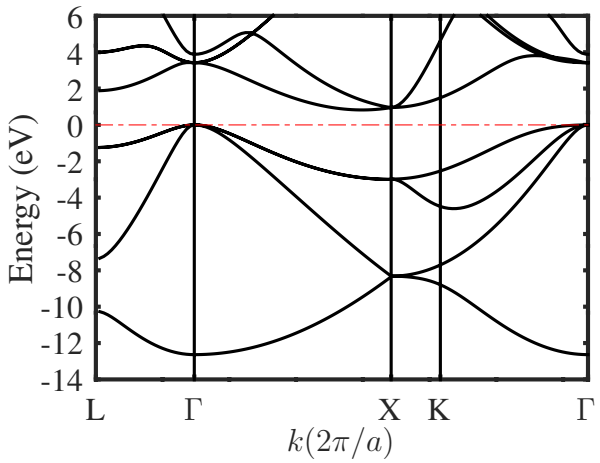


Figure 1. Band structure of Si.

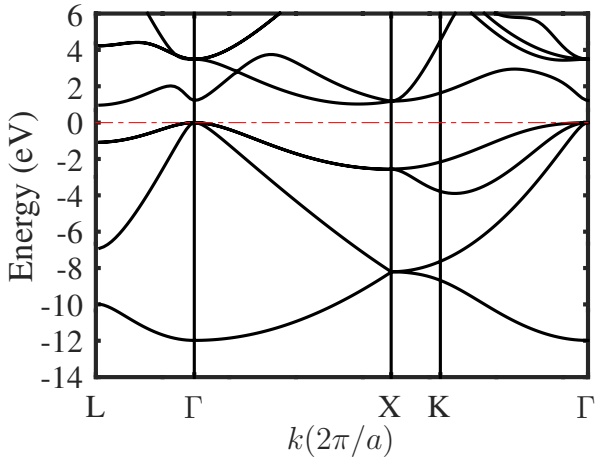


Figure 2. Band structure of Ge.

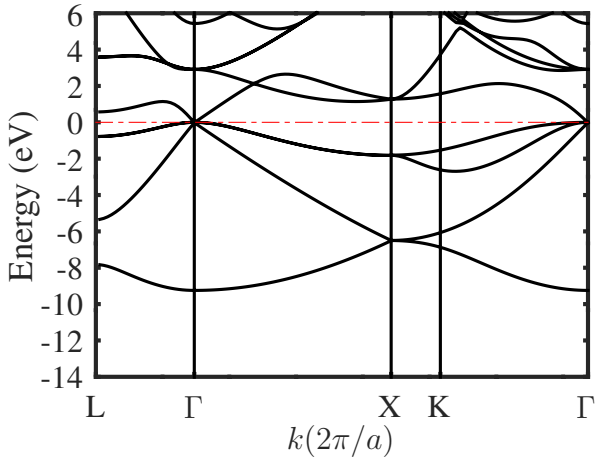


Figure 3. Band structure of Sn.

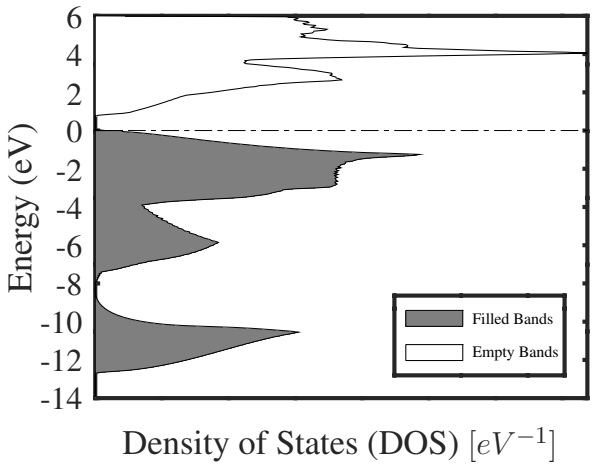


Figure 4. DOS of Si for $q = 80$ in Eq 10 and $\sigma = 0.05$ in Eq 11.

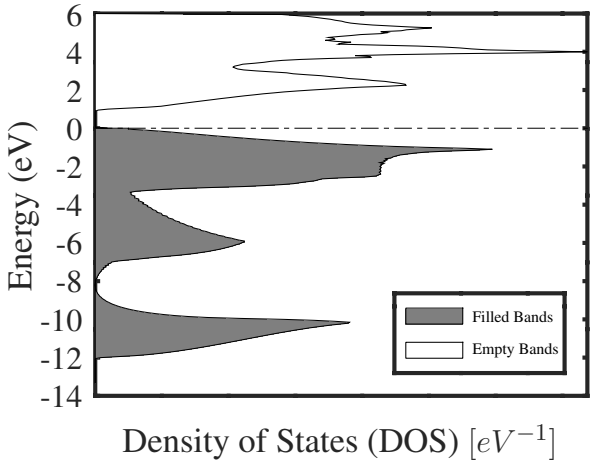


Figure 5. DOS of Ge for $q = 80$ in Eq 10 and $\sigma = 0.05$ in Eq 11.

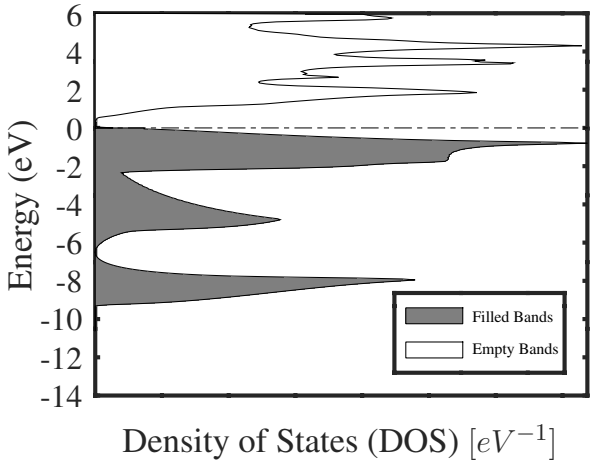


Figure 6. DOS of Sn for $q = 80$ in Eq 10 and $\sigma = 0.05$ in Eq 11.

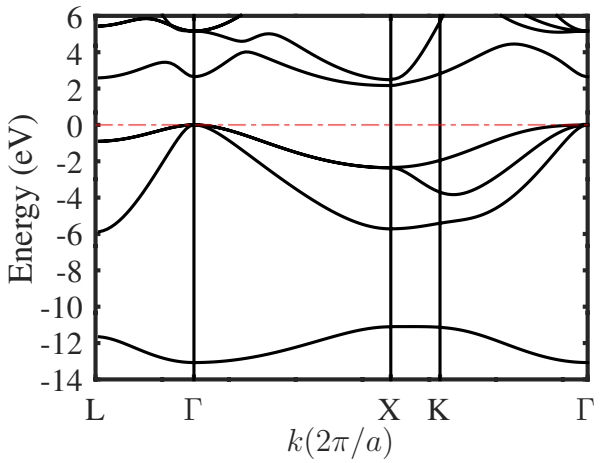


Figure 7. Band structure of GaP.

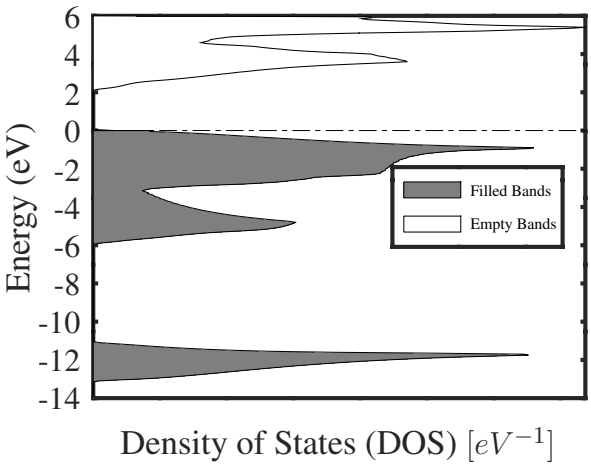


Figure 10. DOS of GaP for $q = 80$ in Eq 10 and $\sigma = 0.05$ in Eq 11.

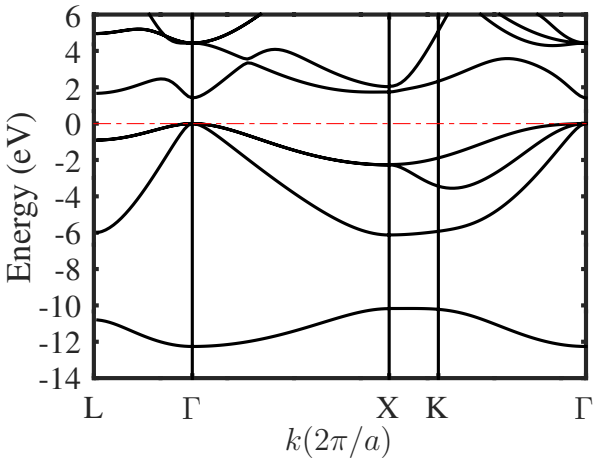


Figure 8. Band structure of GaAs.

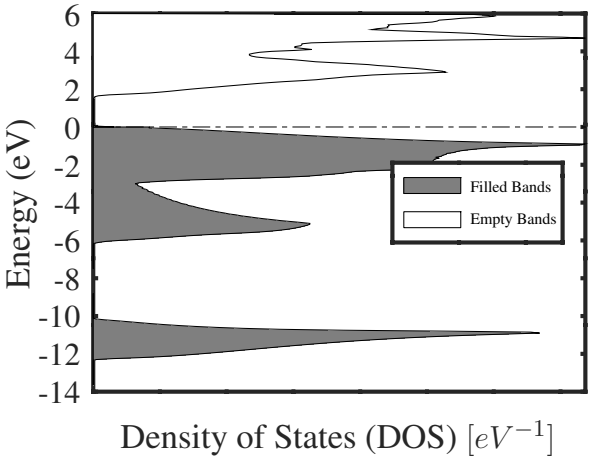


Figure 11. DOS of GaAs for $q = 80$ in Eq 10 and $\sigma = 0.05$ in Eq 11.

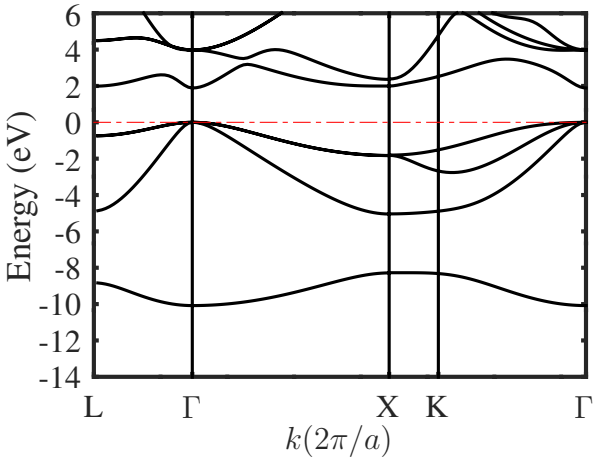


Figure 9. Band structure of AlSb.

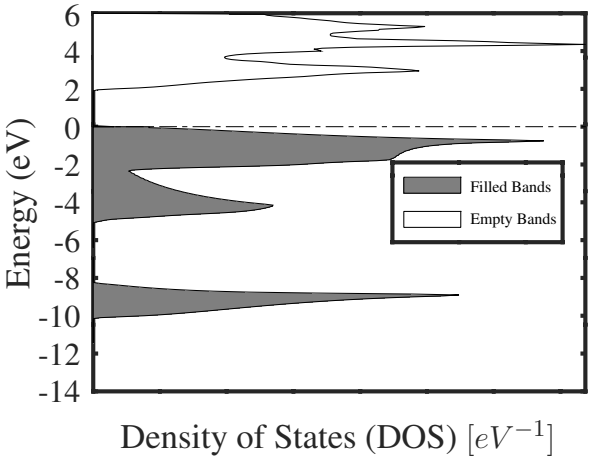


Figure 12. DOS of AlSb for $q = 80$ in Eq 10 and $\sigma = 0.05$ in Eq 11.

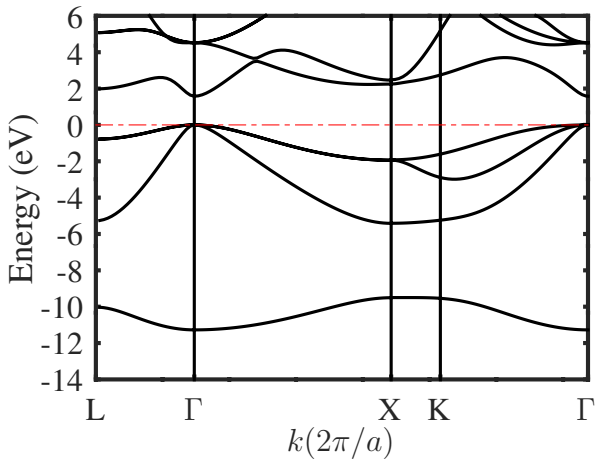


Figure 13. Band structure of InP.

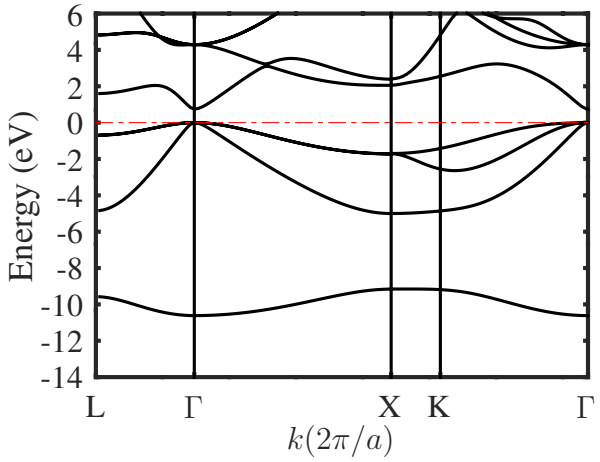


Figure 14. Band structure of GaSb.

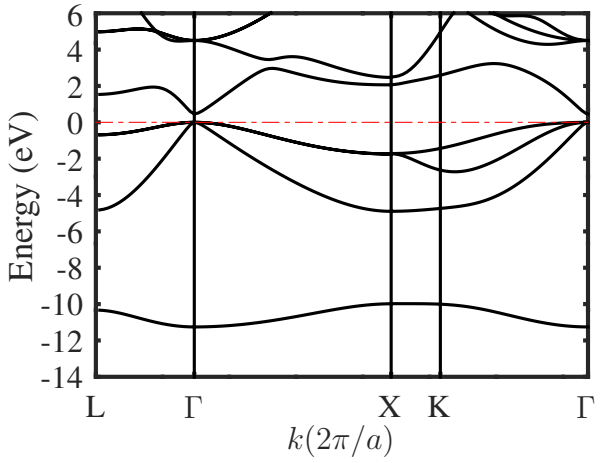


Figure 15. Band structure of InAs.

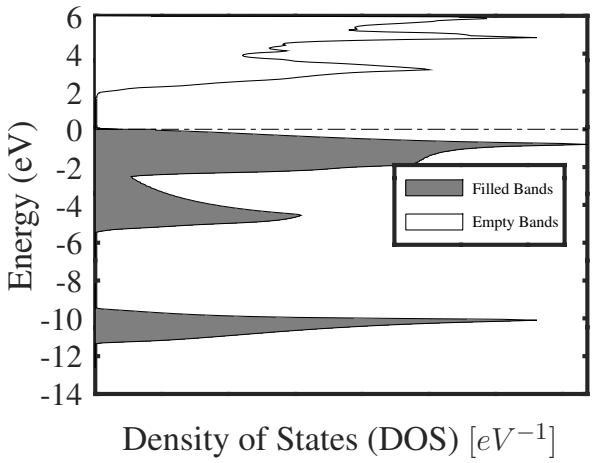


Figure 16. DOS of InP for $q = 80$ in Eq 10 and $\sigma = 0.05$ in Eq 11.

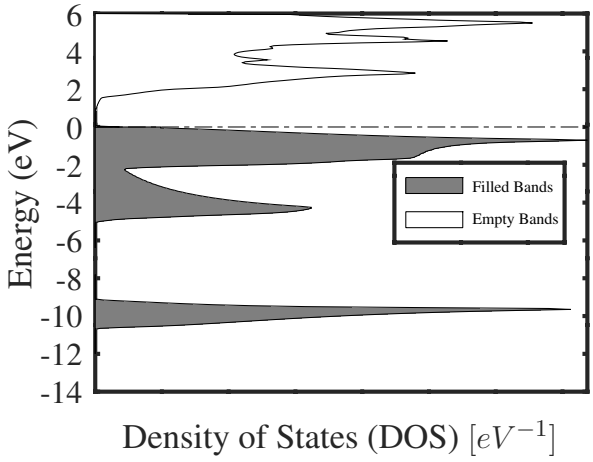


Figure 17. DOS of GaSb for $q = 80$ in Eq 10 and $\sigma = 0.05$ in Eq 11.

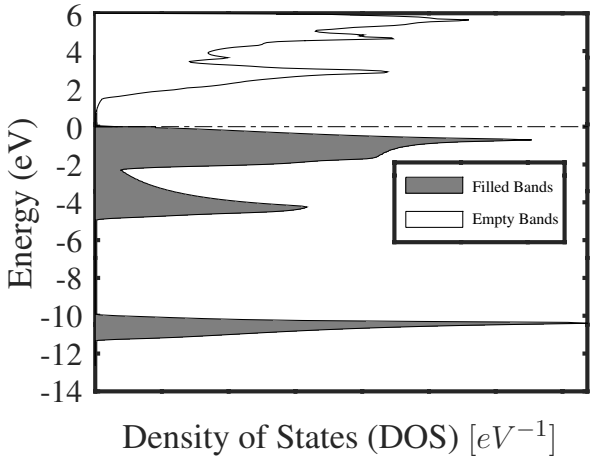


Figure 18. DOS of InAs for $q = 80$ in Eq 10 and $\sigma = 0.05$ in Eq 11.

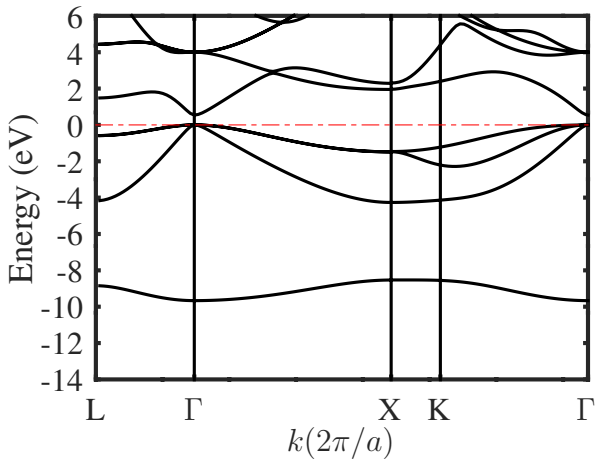


Figure 19. Band structure of InSb.

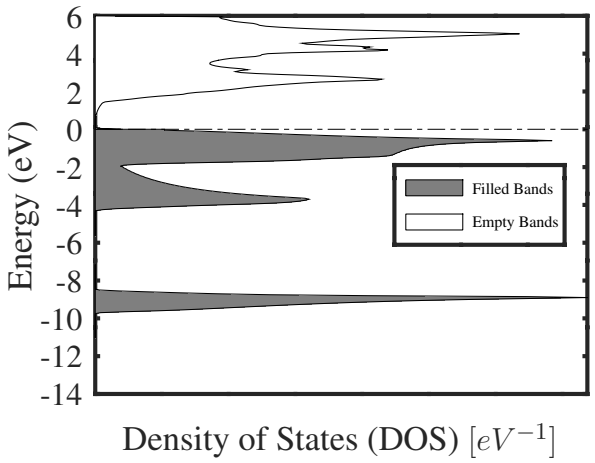


Figure 22. DOS of InSb for $q = 80$ in Eq 10 and $\sigma = 0.05$ in Eq 11.

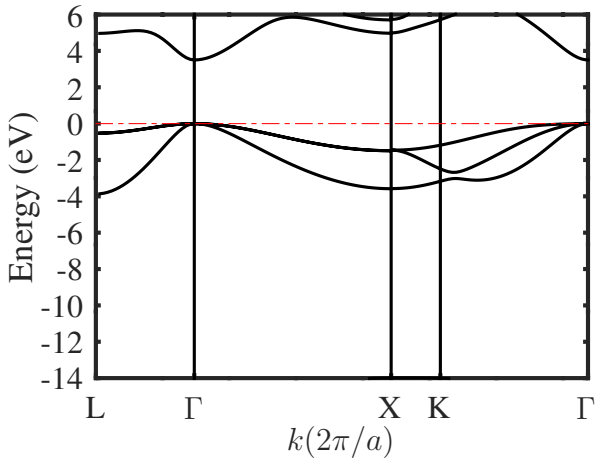


Figure 20. Band structure of ZnS.

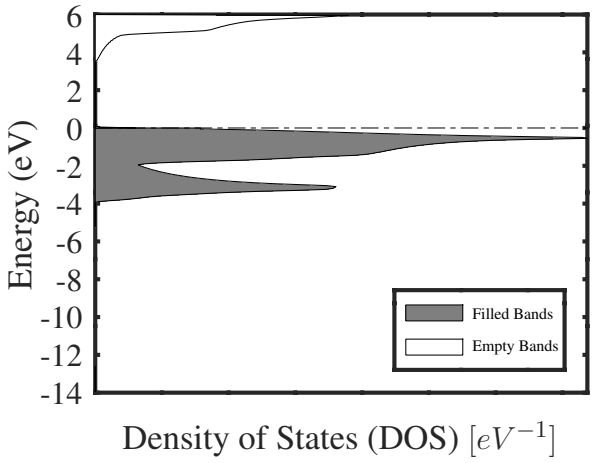


Figure 23. DOS of ZnS for $q = 80$ in Eq 10 and $\sigma = 0.05$ in Eq 11.

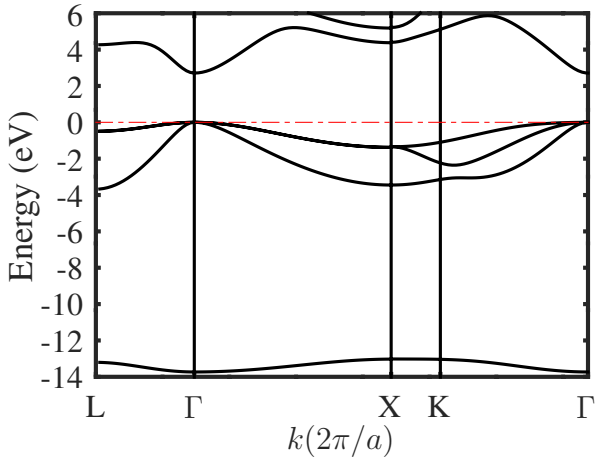


Figure 21. Band structure of ZnSe.

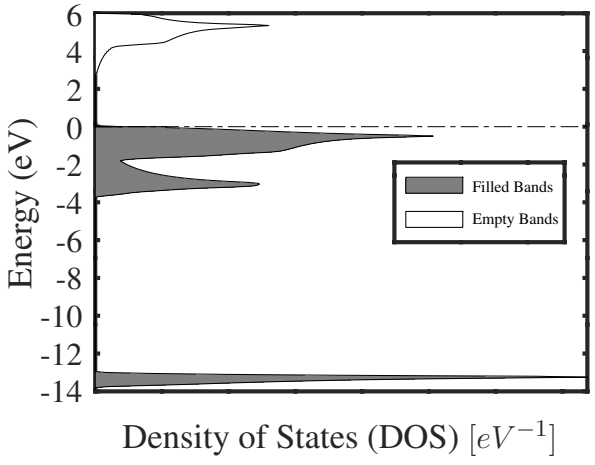


Figure 24. DOS of ZnSe for $q = 80$ in Eq 10 and $\sigma = 0.05$ in Eq 11.

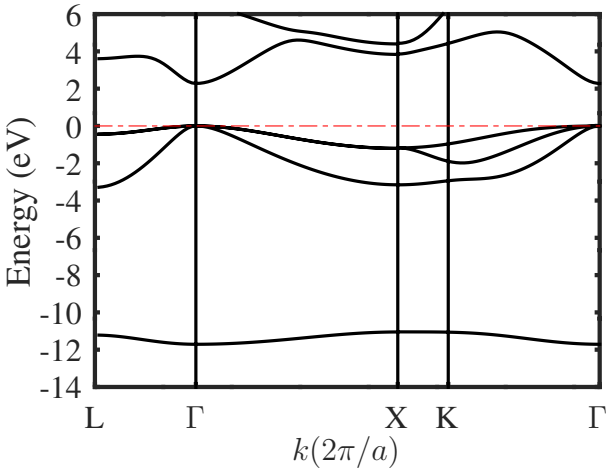


Figure 25. Band structure of ZnTe.

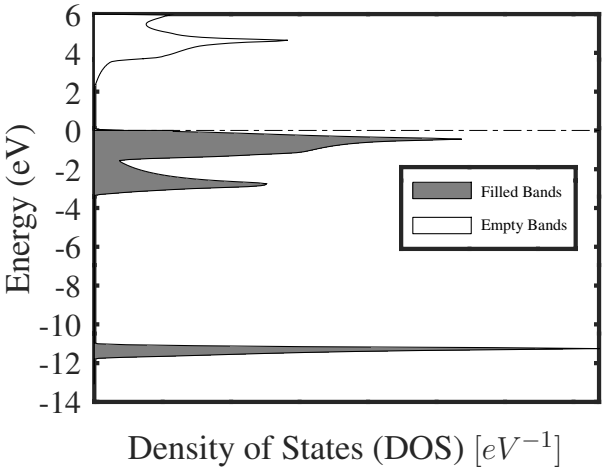


Figure 27. DOS of ZnTe for $q = 80$ in Eq 10 and $\sigma = 0.05$ in Eq 11.

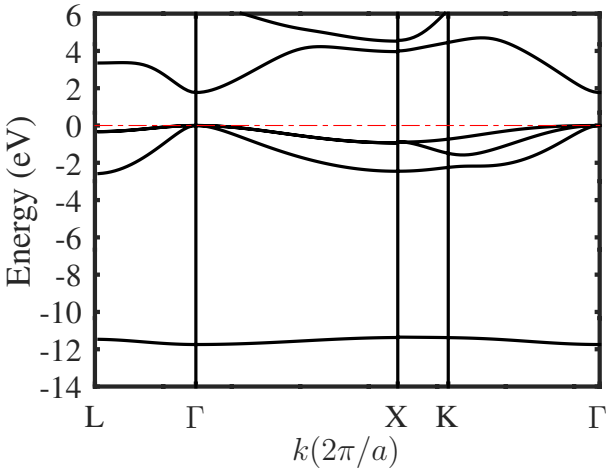


Figure 26. Band structure of CdTe.

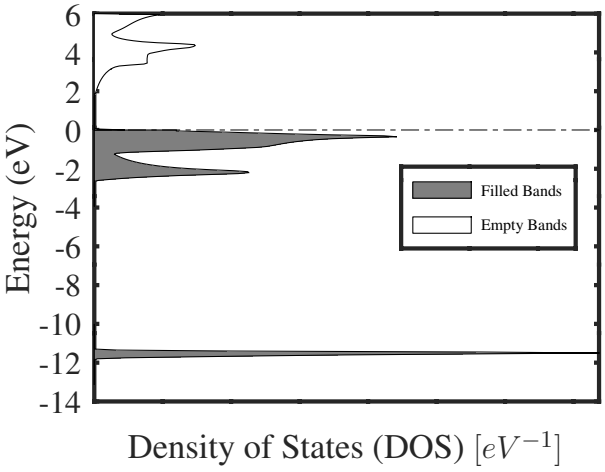


Figure 28. DOS of CdTe for $q = 80$ in Eq 10 and $\sigma = 0.05$ in Eq 11.

IV. Band structures showing all 16 Bands for 14 materials

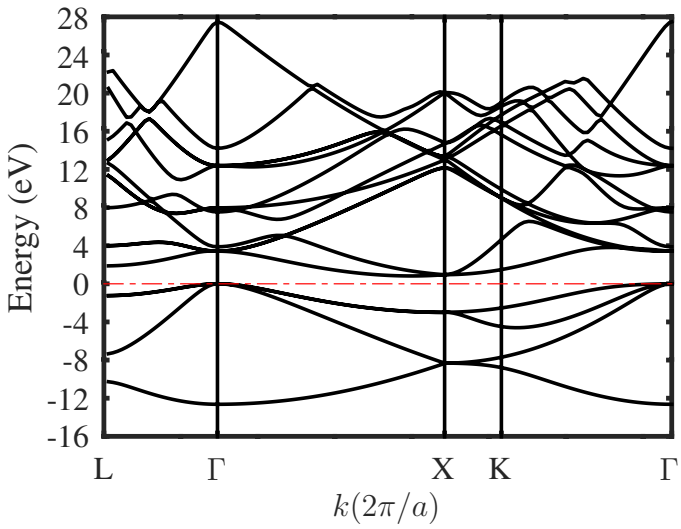


Figure 29. Band structure of Si showing 16 bands.

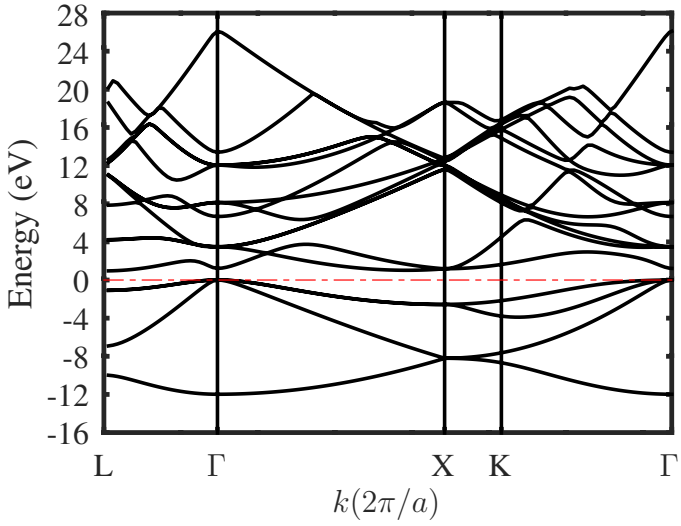


Figure 30. Band structure of Ge showing 16 bands.

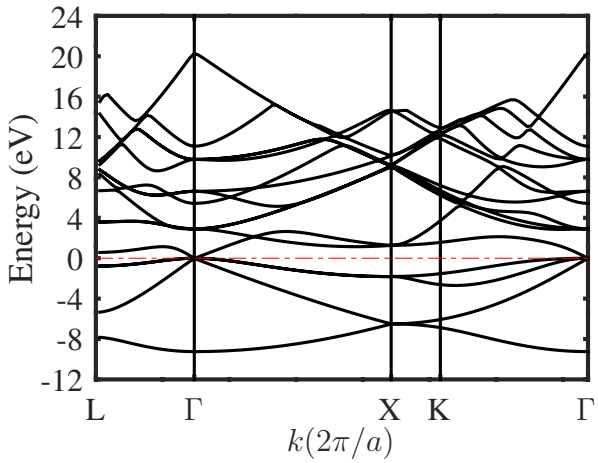


Figure 31. Band structure of Sn showing 16 bands.

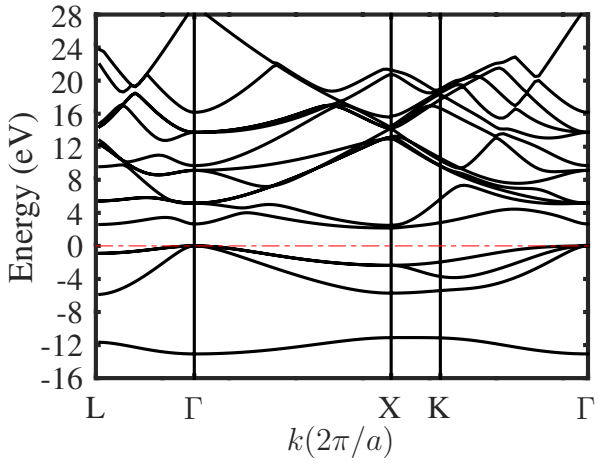


Figure 32. Band structure of GaP showing 16 bands.

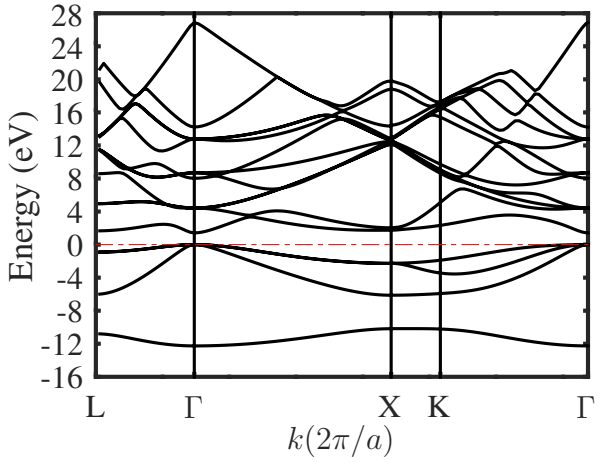


Figure 33. Band structure of GaAs showing 16 bands.

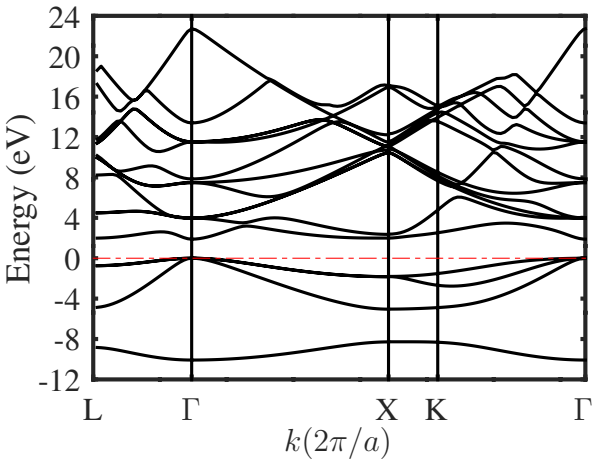


Figure 34. Band structure of AlSb showing 16 bands.

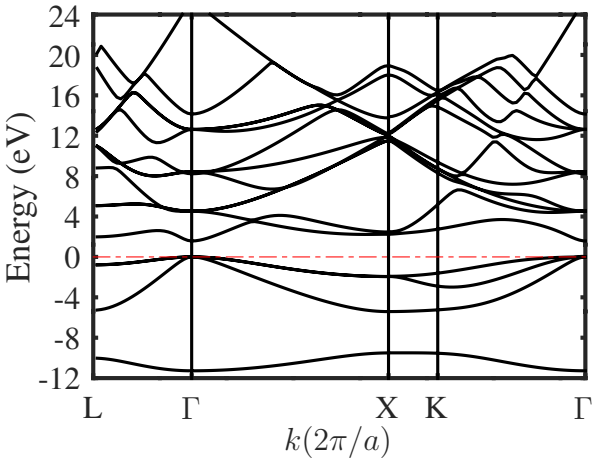


Figure 35. Band structure of InP showing 16 bands.

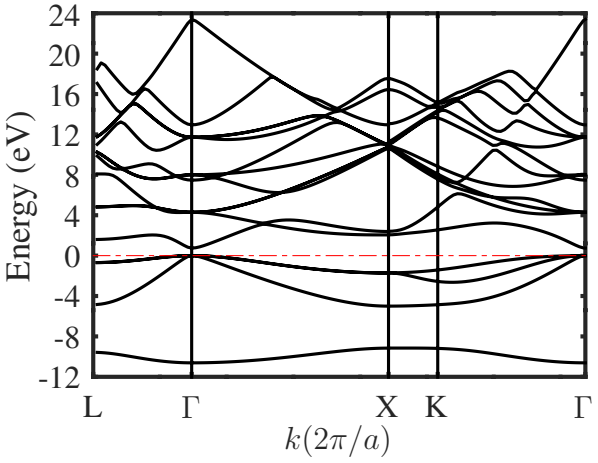
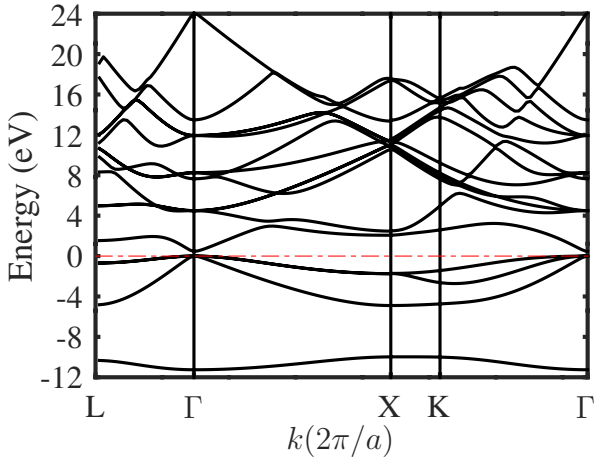


Figure 36. Band structure of GaSb showing 16 bands.



captionBand structure of InAs showing 16 bands.

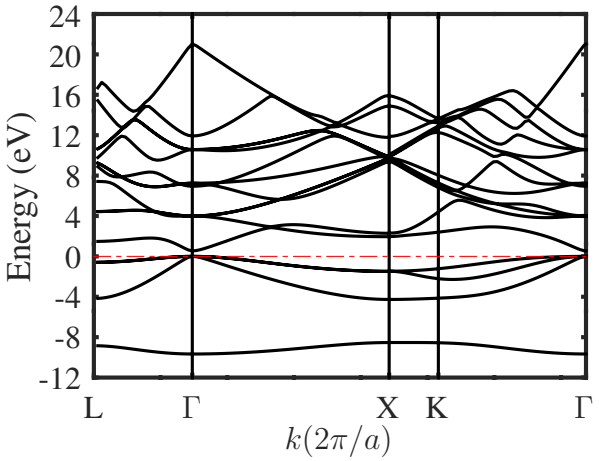


Figure 37. Band structure of InSb showing 16 bands.

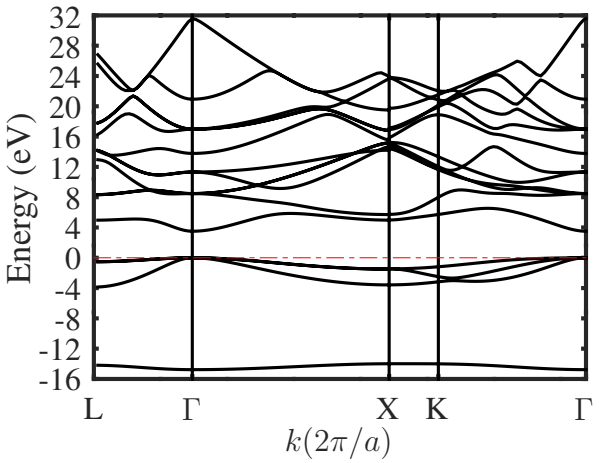


Figure 38. Band structure of ZnS showing 16 bands.

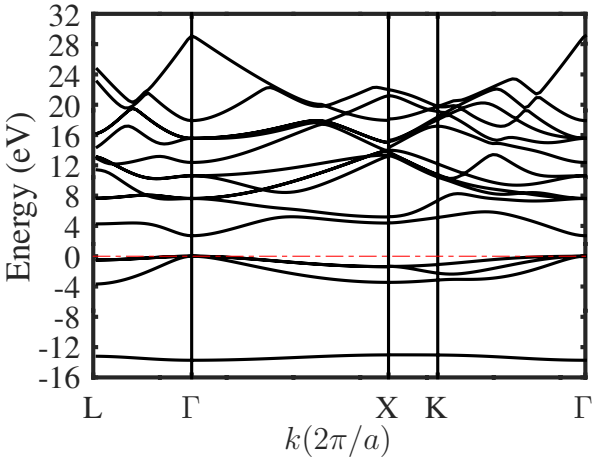


Figure 39. Band structure of ZnSe showing 16 bands.

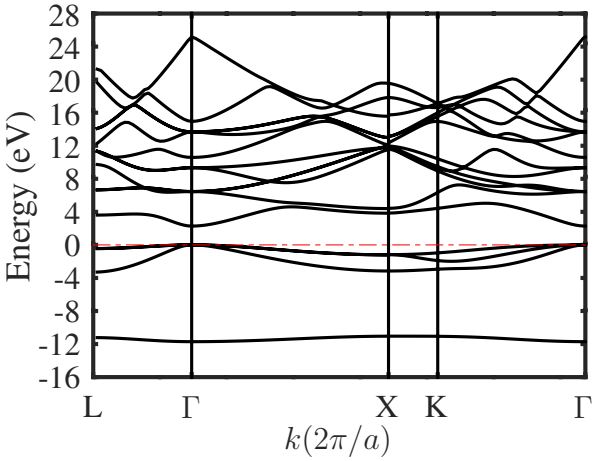


Figure 40. Band structure of ZnTe showing 16 bands.

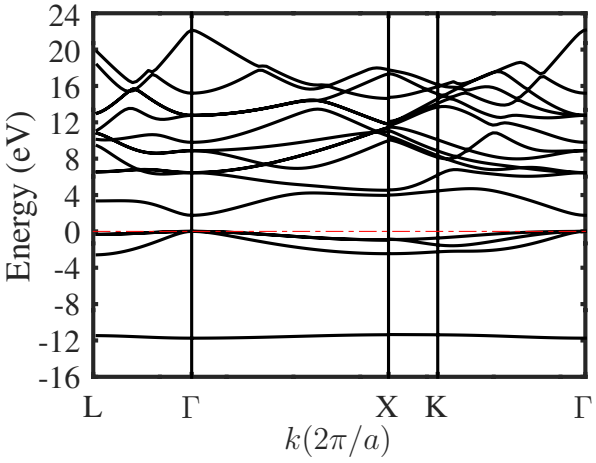


Figure 41. Band structure of CdTe showing 16 bands.

V. Octave Scripts

A. Start Defaults (mystartdefaults.m)

```
%% START DEFAULT COMMANDS OF *.m SCRIPT
clear all;       % Clear all variables/functions in memory
close all force; % Close all figures already opened
clc;             % Clear screen in the command window

iota = sqrt(-1);

%% THE SEVEN EXACT DEFINING CONSTANTS OF THE SI UNIT SYSTEM (2019 UPDATE)
hyperfine = 9192631770;       % Hyperfine transition frequency of Cs-133 [Hz]
celeritas = 299792458;        % Speed of light in vacuum [m/s]
Planck    = 6.62607015E-34;   % Planck's constant [Js]
qel       = 1.602176634E-19;  % Elementary charge (Absolute value of electron charge) [C]
kB        = 1.380640E-23;     % Boltzmann's constant [J/K]
Avogadro  = 6.02214076E-23;   % Avogadro's constant [1/mole]
kcd       = 683;              % Luminous efficacy of 540 THz radiation [candela=lumen/Watt]
                              % Green light at 555,016 nm = maximum possible luminous efficacy.
                              % Originally = peak sensitivity of "average" human eye.

%% PHYSICAL CONSTANTS OF ELECTROMAGNETISM
epsilon0  = 8.8541878128E-12; % Electrical constant (vacuum dielectric permittivity) [F/m]
mu0       = 1.25663706212E-6; % Magnetic constant (vacuum magnetic  permeability) [N/A^2]
                              % close to 4*pi*1.E-7 =1.25663706143E-6
Klitzing  = 25812.80745;      % von Klitzing's constant = Planck/qel^2
                              % respecting significant digits [Ohm]

%% UNITS USED IN ATOMIC, MOLECULAR AND SOLID STATE PHYSICS
hbar      = 1.054571817E-34;                  % Reduced Planck's constant = Planck/(2*pi)
                                              % respecting significant digits [Js]
Angstroem = 1.0E-10;                          % Angstr m [m]
amu       = 1.66053906660E-27;                % Atomic mass unit [kg] = 1 Dalton
                                              % = mass of Carbon_12 atom / 12
elm       = 9.1093837015E-31;                 % Electron mass [kg]
nem       = 1.67492749804E-27;                % Neutron mass [kg]
prm       = 1.67262192369E-27;                % Proton mass [kg]
elecint   = qel^2/(4*pi*epsilon0);            % Scale of electron-electron interaction
Bohr      = hbar^2/(elm*elecint)/Angstroem;   % Bohr radius [Angstr m]
Rydberg   = (elm/(2*hbar^2)) * (elecint)^2/qel; % Rydberg [eV]
Hartree   = 2*Rydberg;                        % Hartree [eV]

%% Units used in calculations of Band Structures

recipunit = 1.0e+10; % Inverse Angstr m
ekin_fact = ((hbar*recipunit)^2/(2 * elm))/qel;

%% WARNING
% The rest of this file can only be executed when sourced in a *.m file
% that includes plotting commands.

%% DEFAULT PLOT CONFIGURATION
% get(groot,'factory')           % List factory-defined plot configurations
% get(groot,'factoryObjectType') % List factory-defined properties for a specific object.
                                 % Examples of 'ObjectType' are: 'Axes', 'Figure', 'Image',
                                 % 'Line','Surface', 'Text', 'ui' (user interface), etc.

% groot is the "handle index" (identifier) of the object
% that is "parent" of all plot objects of the session.

%% THICKER LINES & CHARACTERS FOR VIDEOPROJECTION IN CLASSROOM
set(groot,'defaultLineLineWidth',1);  % Function "set" is not case sensitive.
                                      % Upper cases for
                                      % readability.
set(groot,'defaultAxesFontSize',12);
set(groot,'defaultAxesFontWeight','bold');
set(groot,'defaultAxesLineWidth',2);
set(groot,'defaultAxesXaxisLocation','bottom'); % Location of abcissae ticks & labels
                                                % Possible values:'top','bottom',
                                                % left', 'right', 'origin'.
                                                % 'origin' puts abcissae labels
                                                % on y=0 line

%% DEFAULT COLOR ORDER
% default_colors=get(gca,'colororder'); % gca = "get current axes" = axe system identifier
```

```
% default_colors = default order for coloring curves in last versions of Matlab/Octave:
% (1) Tropical blue, (2) Deep orange, (3) Deep yellow,
% (4) Violet,        (5) Grass green, (6) Azure blue,   (7) Sienna.

%% DEFINING YOUR OWN COLOR ORDER

mycolors=[ % Color order of successive curves must be defined by a 7x3 matrix
0    0    0      % Black
1    0    0      % Red
0    0    1      % Blue
0    0.5  0      % Dark Green
0.9  0.5  0.1    % Orange
0    0.75 0.75   % Turquoise
0.5  0.5  0.5    % Grey
];

%set(groot,'defaultAxesColorOrder',mycolors);

%% TRACING HORIZONTAL AND VERTICAL LINES ACROSS THE PLOT WINDOW

% yline(val) traces horizontal line y(x)=val using current xlim
yline = @(yval, varargin) line(xlim, [yval yval], varargin{:});

% xline(val) traces vertical   line at x=val using current ylim
xline = @(xval, varargin) line([xval xval], ylim, varargin{:});

%% TRACING X=0 AND Y=0 AXES
function z = plotzeros() % Warning: Octave implementation only!
                         % Matlab requires to define this
                         % function in an independent file plotzeros.m
                         % or at the end of the main script
    xline = @(xval, varargin) line([xval xval], ylim, varargin{:});
    yline = @(yval, varargin) line(xlim, [yval yval], varargin{:});
    xline(0,'color',[0 0 0],'linewidth',0.5,'linestyle','-');
    yline(0,'color',[0 0 0],'linewidth',0.5,'linestyle','-');
end

%% END DEFAULT COMMANDS OF *.m SCRIPT
```

B. Brillouin Zone (BZ) Exploration Path

Generate Brillouin Zone (BZ) exploration path according to traditional solid-state representation for FCC. bzpath.dat file generated by this code is used in band structure calculations.

```
clc
clear all
close all
verbose=1;

fprintf('\nGenerate Path in Brillouin Zone (BZ) \n')
fprintf('according to traditional solid state representation.\n')

step= 0.02; % step along path in BZ
s = 4; % number of segments defining path in BZ
qs = zeros(3,s); %initialize start point of segments
qe=zeros(3,s); % initialize end points of segments

qs(1:3,1) = [0.5,0.5,0.5]';       % L = start point 1
qe(1:3,1) = [0 0 0]';             % Gamma = end point 1

qs(1:3,2) = [0 0 0]';       % Gamma : Start point 2
qe(1:3,2) = [1 0 0]';     % X: end point 2

qs(1:3,3) = [1 1 0]';
qe(1:3,3) = [0.75 0.75 0]'; % K : End point 3

qs(1:3,4) = [0.75 0.75 0]';
qe(1:3,4) = [0 0 0]'; % Gamma: end point 4

qu = qe-qs
qu = sign(qu)

q = zeros(5,1); % initialize q vector
p=zeros(5,1); % initialize vector preceeding q

f1 = fopen ('bzpath.dat',"w");
```

```
if (verbose==1)
  printf('iq    is    q(1)     q(2)       q(3)      q(4)     q(5)  \n');
endif
fprintf(f1,'  iq    is           q(1)            q(2)            q(3)            q(4)            q(5)\n');

iq = 0; %initialize number of points for the path in BZ

for is=1:s % loop overs segments
  printf('segmen %d of path in BZ\n', is)
  ip = 0;
  cond=0;
  while(cond !=1)
  xsi = ip*step;
  buf = xsi*qu(:,is);
  cond = all(abs(buf)>=abs((qe(:,is)-qs(:,is)))); % cond =1  if all vector components are zeros if going
    beyond end point
  iq++;
  if(cond==1) % last point if segment
    q = qe(1:3,is);
    q(4)=sqrt(q'*q);
    gap = max(abs(qe(1:3,is)-p(1:3)));
    q(5) = p(5)+gap;
  else % standard point of segment
    q = buf(1:3)+qs(1:3,is);
    q(4) = sqrt(q'*q);
    if(iq==1)
      q(5)=0;
      p(5)=0;
    else
      if(ip==0)
        q(5) = p(5);
      else
        q(5) = p(5) + step;
      endif
    endif
  endif
  p=q;
  if(verbose==1)
    fprintf('%4d %4d %15.6G %15.6G %15.6G %15.6G %15.6G\n',iq, is,q);
  endif
  fprintf(f1, '%4d %4d %15.6G %15.6G %15.6G %15.6G %15.6G\n',iq, is,q)
  ip++;
endwhile
endfor
fclose(f1);
```

C. Code for Band Structure Calculations

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                                                           %
%  This is the main file for Band Structure Plots. The folder which has this main file should have the    %
%  following accompanying files in it. Otherwise program will not work.                                     %
%                                                                                                           %
%    1) mystartdefaults.m                                                                                   %
%    2) bzpath.dat                                                                                          %
%    3) plot3Ddata.m                                                                                        %
%                                                                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


source('mystartdefaults.m');  % Applies default setting from mystartdefaults.m file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Step 1 : Inputs

setpot = 1;                    % Choose one number from 1-14 to select the material from list below
nband=16;                      % No of bands to be stored in output file
cutoff = 21;                   % deal with only |G|^2 < cutoff (2*pi/spacing units)^2 is Hamiltonian
Gs_max = 11;                   % |G|^2  of highest non zero fourier coefficients in expanding potential




%           List of Materials

```

```
ListOfMaterials={"Si","Ge","Sn","GaP","GaAs","AlSb","InP","GaSb","InAs","InSb","ZnS","ZnSe","ZnTe","CdTe"};
MaterialName=char(ListOfMaterials(1,setpot));
fprintf("Selected material is %s.\n\n",MaterialName)

% Potentials from paper in ff (fourier forms) : First columns adjusts the top of filled band to zero
%          V0 VS3  VS8  VS11 VA3 VA4 VA11
ff(1,:) =   [-0.770437 -0.21 0.04 0.08 0 0 0];            % Si
ff(2,:) =   [-0.694179 -0.23 0.01 0.06 0 0 0];            % Ge
ff(3,:) =   [-0.500885 -0.20 0 0.04 0 0 0];               % Sn
ff(4,:) =   [-0.676246 -0.22 0.03 0.07 0.12 0.07 0.02];   % GaP
ff(5,:) =   [-0.651775 -0.23 0.01 0.06 0.07 0.05 0.01];   % GaAs
ff(6,:) =   [-0.509435 -0.21 0.02 0.06 0.06 0.04 0.02];   % AlSb
ff(7,:) =   [-0.561726 -0.23 0.01 0.06 0.07 0.05 0.01];   % InP
ff(8,:) =   [-0.51032 -0.22 0 0.05 0.06 0.05 0.01];       % GaSb
ff(9,:) =   [-0.523957 -0.22 0 0.05 0.08 0.05 0.03];      % InAs
ff(10,:) =  [-0.445297 -0.20 0 0.04 0.06 0.05 0.01];      % InSb
ff(11,:) =  [-0.466304 -0.22 0.03 0.07 0.24 0.14 0.04];   % ZnS
ff(12,:) =  [-0.448188 -0.23 0.01 0.06 0.18 0.12 0.03];   % ZnSe
ff(13,:) =  [-0.391512 -0.22 0.00 0.05 0.13 0.10 0.01];   % ZnTe
ff(14,:) =  [-0.309389 -0.20 0 0.04 0.15 0.09 0.04];      % CdTe

% lattice spacing of all the materials in angstrom
latticespacing = [5.43, 5.66, 6.49, 5.44, 5.64, 6.13, 5.86, 6.12, 6.04, 6.48, 5.41, 5.65, 6.07, 6.41];
spacing = latticespacing(1,setpot); % lattice spacing in angstrom for the selected material


%           Position of atoms in Primitive Cell

tau = zeros(3,2);
tau(:,1) = [0.125 0.125 0.125]';    % position of atom 1 in primitive cell
tau(:,2) = [-0.125 -0.125 -0.125]'; % position of atom 2 in primitive cell

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Step 2: Defining Unit Cell and Cell Volume in Cartesian Coordiante System

fprintf('FCC Lattice unit vectors in Cartesian Coordinates.\n')
a = zeros(3,3);
a(:,1) = [0.5 0.5 0.0]'; % direct lattice unit vector 1
a(:,2) = [0.0 0.5 0.5]'; % direct lattice unit vector 2
a(:,3) = [0.5 0.0 0.5]'; % direct lattice unit vector 3
printf("\n      a_1      a_2      a_3\n")
disp(a)

cell_volume = a(:,1)' * cross(a(:,2),a(:,3))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Step 3: Defining Reciprocal Lattice Vectors

g=zeros(4,3);

g(1:3,1) = cross(a(:,2),a(:,3))/cell_volume;
g(1:3,2) = cross(a(:,3),a(:,1))/cell_volume;
g(1:3,3) = cross(a(:,1),a(:,2))/cell_volume;

fprintf("\n\nFCC Reciprocal lattice unit vectors in Cartesian Coordinates.\n\n")
for i =1:3
  g(4,i) = g(1:3,i)' * g(1:3,i);
end
printf(" g_1 g_2 g_3\n")
disp(g)
min_norm = sqrt(min(g(4,:))) % minimal norm
nstep = floor(sqrt(cutoff)/min_norm) + 1;   %  Number of positive steps along each reciprocal lattice unit vector
printf("\n\nCutoff requires %d positive steps along each reciprocal lattice uniit vector.\n",nstep);

nodes = (2*nstep + 1)^3;  % Number of the reciprocal lattice vectors generated
printf("\n\nGenerate (2*%1d +1)^3 = %4d reciprocal lattice vectors\n\n",nstep,nodes);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Step 4: Generating reciprocal lattice vectors in all directions for calculations
G = zeros(5,nodes);
n=0;
for j = -nstep:nstep
  for k = -nstep:nstep
    for l = -nstep:nstep    % small L is iterating
```

```
      n++;
      G(1:3,n) = j*g(1:3,1)+k*g(1:3,2)+l*g(1:3,3);
      G(5,n) = G(1:3,n)'*G(1:3,n);
      G(4,n) = sqrt(G(5,n));
    end
  end
end

GT = sortrows(G',4); % sorting reciprocal lattice vectors by growing norm
G = GT';

kept = 1;
for n = 2:nodes
  if(G(5,n) <= cutoff)
    kept++;
  end
end

printf("%4d G vectors featuring |G|^2<cutoff\n",kept) % you should get 113 here
printf("  n              G(1)              G(2)              G(3)              |G|\n");
for i = 1:kept
printf("%3.6G %15.6G  %15.6G  %15.6G  %15.6G\n",i,G(1,i),G(2,i),G(3,i),G(4,i));
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Step 5: Loading Brillouin Zone (bzpath.dat) Data File

datafile= 'bzpath.dat'; % 3D Brillouin zone path Data file name
specify_format = 'yes';
delim_in = ' ';
head_in = 1;

if (strcmp(specify_format,'yes'))
  [x, delim_out, head_out] = importdata(datafile, delim_in, head_in);
else
  [x,delim_out,head_out] = importdata(datafile);
end

sz = size(x.data);
nq = sz(1);       % No of q vectors
for iq = 1:nq
  is(iq) = x.data(iq,2);
  q(iq,1:5) = x.data(iq,3:7);
end

if head_out > 0
  x = x.data;
else
  x = x;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Step 6: For the kept reciprocal lattice vectors, calculate the value of V_G defined in Eq 4 (report)
%                                                                                                   Adopted from Monkhorst


if (setpot!=0) % fourier components of empty lattice potential
  ekinunit = 1.;
  printf("\nEnergy in (hbar^2/2*elm)*(2*pi/a)^2 units\n");
  for n=1:kept
    cvg(n) = 0+0i;
  end
end

if (setpot!=0) % fourier components of selected material pseudopotential
  printf("\nenergy in eV\n");
  ekinunit = ekin_fact*(2.d0*pi/spacing)^2 % kinectic energy in EV
  for n=1:kept
    sym = 0.;
    asym=0.;
    if(G(5,n) == 0)
```

```
      sym =ff(setpot,1)*Rydberg ; % here you can adjust zero of the potential to top of the valence band
      asym =0.;
    end
    if (G(5,n) == 3) % if (abs(G(5,n)-3)<tol)
      sym = ff(setpot,2)*Rydberg;
      asym = ff(setpot,5)*Rydberg;
    end
    if (G(5,n) == 4)
      sym =0.;
      asym = ff(setpot,6)*Rydberg;
    end
    if (G(5,n) == 8)
      sym = ff(setpot,3)*Rydberg;
      asym = 0.;
    end
    if (G(5,n) == 11)
      sym = ff(setpot,4)*Rydberg;
      asym =ff(setpot,7)*Rydberg;
    end
  argu = 2*pi * (G(1:3,n)'*tau(1:3,1));
  cvg(n) = cos(argu)*sym - 1i*sin(argu)*asym; % caution: sign of Im part
  end
end

printf("%4d G vectors featuring |G|^2<cutoff\n",kept) % you should get 113 here
  printf("  n             G(1)             G(2)             G(3)             |G|           |G|^2      Real Values   Imaginary Values\n");
for i = 1:kept
  printf("%3.6G %15.6G  %15.6G  %15.6G  %15.6G %15.6G %15.6G %15.6G \n",i,G(1,i),G(2,i),G(3,i),G(4,i),abs(G(4,i))^2,real(cvg(i)),imag(cvg(i)));
end

MyFileName=sprintf(['E',num2str(setpot),MaterialName,"-3D-EK-Diagram.dat"]); % Writing to a data files named with material
fprintf("\n\nName of the data file stored in computer for %s is: %s\n\n",MaterialName,MyFileName)
f2 = fopen(MyFileName,'w');

fprintf('\n Diagonalization loop over %4d wavevectors:',nq);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Step 7: Initialize Hamiltonian matrix and assign potential energy values

G_diff = zeros(5,1);
H=zeros(kept,kept); % initialization of the Hamiltonian matrix
  %potential energy % you can place this potential energy part before the loop as optimization
  for j =1:kept
    for i =1:kept
      G_diff(1:3) = G(1:3,i) - G(1:3,j);
      G_diff(5) = G_diff(1:3)'*G_diff(1:3);
      if(G_diff(5) <= Gs_max)
      for k = 1:kept
        % if (abs(G_diff(1:3) - G(1:3,k))<[tol tol tol]')
        % would be better when comparing non-integer values
        if ((G_diff(1:3) - G(1:3,k)) == [0 0 0]')
          H(i,j) = cvg(k);
        end
      end
    end
  end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Step 8: Calculate the difference |k     G|^2 along the BZ exploration path and kinetic energy part of
%                   the hamiltonian matrix then diagonalize to find the eigenenergies

for iq =1:nq
  fprintf('%4d',iq); % This is counter to show on Command windows
  % Kinetic Energy
  for i = 1:kept
    for m=1:3
      p(m) = q(iq,m) - G(m,i);
    end
    H(i,i) = ekinunit * (p*p') + ff(setpot,1)*Rydberg; % + Adjusting top of the filled band to zero
  end
```

```

  % Hermiticity Check
  tol = 1e-10;
  if(!ishermitian(H,tol))
    printf("\nHamiltonian matrix not Hermitian : fatal error.\n");
    return;
  else
    % Diagonalization of Hamiltonian
    [v,ev]=eig(H) ;         % Diagonalization [eigenvectors, eigenvalues]
    E = real(diag(ev));     % Hermitian matrix features real eigenvalues
    [E,perm] = sort(E);     % Sorting eigenvalues in increasing order
    v = v(:,perm);          % Re-order eigenvectors in same order as permuted eigenvalues

    % Writing to file for band plotting
    bzs = 100 * (-1)^is(iq);
    if (kept < nband)
      nband = kept;
    end
    fprintf(f2, "%15.6G  %15.6G", q(iq,5), bzs);
    for i = 1:nband
      fprintf(f2,"%15.6G",E(i));
    end
    fprintf(f2, "\n");
    if (q(iq,4) == 0)
      for i=1:nband
        gamma(i) = E(i);
      end
    end
  end
end

% Writing the eigenenergies to a file
fclose(f2);
fprintf('\n\n%2d lowest eigenvalues at Gamma point: \n',nband);
for i = 1:nband
    fprintf('%4d %15.6G\n', i, gamma(i));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Step 9: Plot the eigenenergies against the BZ exploration path to obtain band structure
source('BS_Plot.m')
```

D. Code for Band Structure Plotting (BS_Plot.m)

```
datafile= ['E',num2str(setpot),MaterialName,"-3D-EK-Diagram.dat"]; % Reading data file for specific material

specify_format= 'yes'; % If yes, the two next definitions will be used
delim_in= ' ';         % Expected column separator character
head_in =  1;          % Expected number of lines of header

labelx = 'Wavevector [2\pi/a units]'; % NB: Greek characters in TeX encoding
labely = 'Energy [eV]';

if(strcmp(specify_format,'yes'))
  [z,delim_out,head_out]=importdata(datafile,delim_in,head_in);
else
  [z,delim_out,head_out]=importdata(datafile);
end

fprintf('\nFOUND IN DATA FILE:\n');
fprintf('Column separator character = ''%s''\n',delim_out);
fprintf('Number of lines of header  = %d\n',head_out);
if(head_out>0)
  x=z.data; % If header found, file content read as "structure array"
            % (Display z to see what is meant)
else
  x=z;      % If header not found, file content read as numerical array
end

sz=size(x);
fprintf('abscissae = %d\n',sz(1))
fprintf('columns   = %d\n',sz(2))
myplot = figure()
plot(x(:,1),x(:,2:end),'linewidth',1.5,'k'); % Remember: Sourced plot defaults will apply!
yline(0,"linestyle", "-.",'color','r','linewidth',0.5)
hold on;
```

```
ylim([-14,6]);

set(gca,'xtick',[0 0.5 1.5 1.75 2.5]);
set(gca, 'xticklabel',({"L","$\Gamma$","X","K","$\Gamma$"}));
set(gca, 'ytick', [-14:2:6])
##mytitle=["Band Structure of ",MaterialName];
##title(mytitle,"interpreter","latex")
xlabel("$k(2\pi/a)$","interpreter","latex")
ylabel("Energy (eV)","interpreter","latex")

         %           Next lines must come after calling plot()


        % OUTPUT FILE NAME
##
##dot=rindex(datafile, '.');      % Position of filetype extension delimiter
##root=substr(datafile,1,dot-1);  % Filename without filetype extension
##outfile=sprintf('%s.pdf',root);
##
##         %OUTPUT FILES
##print(outfile,'-dpdf');                % Basic pdf output
##print(outfile,'-dpdflatexstandalone'); % Combined pdf & LaTeX files
##
##fprintf('\nOUTPUT FILES: \n');
##fprintf('%s (basic pdf file)\n',outfile);
##fprintf('%s.tex and %s-inc.pdf (for perfect LaTeX processing)\n',root,root);

waitfor(g); % Wait for closing graphical window
```

E. Code for Density of States (DOS) Calculations

```
%% This is the main file. The folder which has this main file should have the following accompanying files in it. Otherwise
## program will not work. This is rather faster as for loops in DOS calculations have been replaced by matrix operations.

%%    1)  mystartdefaults.m
%%    2)
%%    3)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                                                         %
%  This is the main file for Band Structure Plots. The folder which has this main file should have the    %
%  following accompanying files in it. Otherwise program will not work.                                   %
%                                                                                                         %
%    1) mystartdefaults.m                                                                                 %
%    2)                                                                                         %
%    3)                                                                                       %
%                                                                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

source('mystartdefaults.m');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Step 1 : Inputs

Setpot = 11;     % Choose one number from 1-14 to select the material from list below
MQ_r = 5;       % # of q vectors to generate in each direction; total being MQ_r^3; Ref: Monkhorst
Smear=0.15;     % Width of Gaussian for DOS Calculation

nband=16;       % No of bands to be stored in output file
cutoff = 21;    % deal with only |G|^2 < cutoff (2*pi/spacing units)^2 is Hamiltonian
Gs_max = 11;    % |G|^2  of highest non zero fourier coefficients in expanding potential



%           List of Materials

ListOfMaterials={"Si","Ge","Sn","GaP","GaAs","AlSb","InP","GaSb","InAs","InSb","ZnS","ZnSe","ZnTe","CdTe"};
MaterialName=char(ListOfMaterials(1,Setpot));
fprintf("Selected material is %s.\n\n",MaterialName)

% Potentials from paper in ff (fourier forms) : First columns adjusts the top of filled band to zero
%          V0 VS3  VS8  VS11 VA3 VA4 VA11
ff(1,:) =   [-0.770437 -0.21 0.04 0.08 0 0 0];            % Si
ff(2,:) =   [-0.694179 -0.23 0.01 0.06 0 0 0];            % Ge
ff(3,:) =   [-0.500885 -0.20 0 0.04 0 0 0];               % Sn
ff(4,:) =   [-0.676246 -0.22 0.03 0.07 0.12 0.07 0.02];   % GaP
```

```
ff(5,:) =   [-0.651775 -0.23 0.01 0.06 0.07 0.05 0.01];   % GaAs
ff(6,:) =   [-0.509435 -0.21 0.02 0.06 0.06 0.04 0.02];   % AlSb
ff(7,:) =   [-0.561726 -0.23 0.01 0.06 0.07 0.05 0.01];   % InP
ff(8,:) =   [-0.51032 -0.22 0 0.05 0.06 0.05 0.01];       % GaSb
ff(9,:) =   [-0.523957 -0.22 0 0.05 0.08 0.05 0.03];      % InAs
ff(10,:) =  [-0.445297 -0.20 0 0.04 0.06 0.05 0.01];      % InSb
ff(11,:) =  [-0.466304 -0.22 0.03 0.07 0.24 0.14 0.04];   % ZnS
ff(12,:) =  [-0.448188 -0.23 0.01 0.06 0.18 0.12 0.03];   % ZnSe
ff(13,:) =  [-0.391512 -0.22 0.00 0.05 0.13 0.10 0.01];   % ZnTe
ff(14,:) =  [-0.309389 -0.20 0 0.04 0.15 0.09 0.04];      % CdTe

% lattice spacing of all the materials in angstrom
latticespacing = [5.43, 5.66, 6.49, 5.44, 5.64, 6.13, 5.86, 6.12, 6.04, 6.48, 5.41, 5.65, 6.07, 6.41];
spacing = latticespacing(1,Setpot); % lattice spacing in angstrom for the selected material


%           Position of atoms in Primitive Cell

tau = zeros(3,2);
tau(:,1) = [0.125 0.125 0.125]';    % position of atom 1 in primitive cell
tau(:,2) = [-0.125 -0.125 -0.125]'; % position of atom 2 in primitive cell

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Step 2: Defining Unit Cell and Cell Volume in Cartesian Coordiante System

fprintf('FCC Lattice unit vectors in Cartesian Coordinates.\n')
a = zeros(3,3);
a(:,1) = [0.5 0.5 0.0]'; % direct lattice unit vector 1
a(:,2) = [0.0 0.5 0.5]'; % direct lattice unit vector 2
a(:,3) = [0.5 0.0 0.5]'; % direct lattice unit vector 3
printf("\n      a_1      a_2      a_3\n")
disp(a)

cell_volume = a(:,1)' * cross(a(:,2),a(:,3))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Step 3: Defining Reciprocal Lattice Vectors

g=zeros(4,3);

g(1:3,1) = cross(a(:,2),a(:,3))/cell_volume;
g(1:3,2) = cross(a(:,3),a(:,1))/cell_volume;
g(1:3,3) = cross(a(:,1),a(:,2))/cell_volume;

fprintf("\n\nFCC Reciprocal lattice unit vectors in Cartesian Coordinates.\n\n")
for i =1:3
  g(4,i) = g(1:3,i)' * g(1:3,i);
end
printf(" g_1 g_2 g_3\n")
disp(g)
min_norm = sqrt(min(g(4,:))) % minimal norm
nstep = floor(sqrt(cutoff)/min_norm) + 1; %  Number of positive steps along each reciprocal lattice unit vector
printf("\n\nCutoff requires %d positive steps along each reciprocal lattice uniit vector.\n",nstep);
nodes = (2*nstep + 1)^3;

printf("\n\nGenerate (2*%1d +1)^3 = %4d reciprocal lattice vectors\n\n",nstep,nodes);




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Step 4: Generating reciprocal lattice vectors in all directions for calculations

G = zeros(5,nodes);
n=0;
for j = -nstep:nstep
  for k = -nstep:nstep
    for l = -nstep:nstep    % small L is iterating
      n++;
      G(1:3,n) = j*g(1:3,1)+k*g(1:3,2)+l*g(1:3,3);
      G(5,n) = G(1:3,n)'*G(1:3,n);
      G(4,n) = sqrt(G(5,n));
    end
  end
```

```
end

GT = sortrows(G',4); % sorting direct lattice vectors by growing norm
G = GT';

kept = 1;

for n = 2:nodes
  if(G(5,n) <= cutoff)
    kept++;
  end
end

printf("%4d G vectors featuring |G|^2<cutoff\n",kept) % you should get 113 here
printf("  n             G(1)             G(2)             G(3)             |G|\n");

for i = 1:kept
printf("%3.6G %15.6G  %15.6G  %15.6G  %15.6G\n",i,G(1,i),G(2,i),G(3,i),G(4,i));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Step 4: Generating reciprocal lattice points in all 8 octants for DOS calculations
%                   Ref: Monkhorst

r = [1:1:MQ_r];
u = (2*r - MQ_r -1)/(2 * MQ_r);   % a uniform sequence: Monkhorst Eq 3

K_Points = zeros(ceil(MQ_r^3/8), 4);     % uniform distribution of K_Points vectors in the first octant only
n = 0;
for j = 1:ceil(MQ_r/2);
    for k = 1:ceil(MQ_r/2);
        for l = 1:ceil(MQ_r/2);
            n++;
            x = u(j) * g(1:3,1) + u(k) * g(1:3,2) + u(l) * g(1:3,3);
            norm_x = sqrt(x' * x);
            K_Points(n,1:3) = x;
            K_Points(n,4) = norm_x;
        end
    end
end

% Generate points in other octants using symmetry
K_Points = [K_Points;
    -K_Points(:,1:3) K_Points(:,4);
     K_Points(:,1:2) -K_Points(:,3) K_Points(:,4);
     K_Points(:,1) -K_Points(:,2:3) K_Points(:,4);
    -K_Points(:,1) K_Points(:,2:3) K_Points(:,4);
    -K_Points(:,1) K_Points(:,2) -K_Points(:,3) K_Points(:,4);
    -K_Points(:,1:2) K_Points(:,3) K_Points(:,4);
     K_Points(:,1) -K_Points(:,2) K_Points(:,3) K_Points(:,4)];


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Step 6: For the kept reciprocal lattice vectors, calculate the value of V_G defined in Eq 4 (report)
%                                                                                        Adopted from Monkhorst


if (Setpot!=0) % fourier components of empty lattice potential
  ekinunit = 1.;
  printf("\nEnergy in (hbar^2/2*elm)*(2*pi/a)^2 units\n");
  for n=1:kept
    cvg(n) = 0+0i;
  end
end

if (Setpot!=0) % fourier components of selected material pseudopotential
  spacing = latticespacing(1,Setpot); % lattice spacing in angstrom for the selected material
  ekinunit = ekin_fact*(2.d0*pi/spacing)^2 % kinectic energy in EV
  printf("\nenergy in eV\n");
  for n=1:kept
    sym = 0.;
    asym=0.;
```

```
    if(G(5,n) == 0)
    sym =ff(Setpot,1)*Rydberg ; % here you can adjust zero of the potential to top of the valence band
    asym =0.;
  end
  if (G(5,n) == 3) % if (abs(G(5,n)-3)<tol)
    sym = ff(Setpot,2)*Rydberg;
    asym = ff(Setpot,5)*Rydberg;
  end
  if (G(5,n) == 4)
    sym =0.;
    asym = ff(Setpot,6)*Rydberg;
  end
  if (G(5,n) == 8)
    sym = ff(Setpot,3)*Rydberg;
    asym = 0.;
  end
  if (G(5,n) == 11)
    sym = ff(Setpot,4)*Rydberg;
    asym =ff(Setpot,7)*Rydberg;
  end
  argu = 2*pi * (G(1:3,n)'*tau(1:3,1));
  cvg(n) = cos(argu)*sym - 1i*sin(argu)*asym; % caution: sign of Im part

end
end


MyFileName=sprintf([MaterialName,"-EigenMatrix.dat"]); % Writing to a data files named with material
fprintf("\n\nName of the data file stored in computer for %s is: %s\n\n",MaterialName,MyFileName)
f2 = fopen(MyFileName,'w');

%fprintf('\n Diagonalization loop over %4d wavevectors:',nq);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Step 7: Initialize Hamiltonian matrix and assign potential energy values


G_diff = zeros(5,1);
H=zeros(kept,kept); % initialization of the Hamiltonian matrix
  %potential energy % you can place this potential energy part before the loop as optimization
  for j =1:kept
    for i =1:kept
      G_diff(1:3) = G(1:3,i) - G(1:3,j);
      G_diff(5) = G_diff(1:3)'*G_diff(1:3);
      if(G_diff(5) <= Gs_max)
      for k = 1:kept
        % if (abs(G_diff(1:3) - G(1:3,k))<[tol tol tol]')
        % would be better when comparing non-integer values
        if ((G_diff(1:3) - G(1:3,k)) == [0 0 0]')
          H(i,j) = cvg(k);
        end
      end
    end
  end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Step 8: Calculate the difference |k     G|^2 along the BZ exploration path and kinetic energy
    part of
%                   the hamiltonian matrix then diagonalize to find the eigenenergies

nq = MQ_r^3;
Energy_Array = [-14:0.05:6];
DOS=zeros(length(Energy_Array));
fprintf("\nCalculating Enery Eigenvalues. Please wait..")

for iq =1:nq
  %fprintf('%8d ',iq); % This is counter to show on Command windows
  % Kinetic Energy
  for i = 1:kept
    for m=1:3
      p(m) = K_Points(iq,m) - G(m,i);
    end
```

```
    H(i,i) = ekinunit * (p*p') + ff(Setpot,1)*Rydberg;
  end

  % Hermiticity Check
  tol = 1e-10;
  if(!ishermitian(H,tol))
    printf("\nHamiltonian matrix not Hermitian : fatal error.\n");
    return;
  else
    % Diagonalization of Hamiltonian
    [v,ev]=eig(H) ;          % Diagonalization [eigenvectors, eigenvalues]
    E = real(diag(ev));      % Hermitian matrix features real eigenvalues
    [E,perm] = sort(E);      % Sorting eigenvalues in increasing order
    v = v(:,perm);           % Re-order eigenvectors in same order as permuted eigenvalues

    % Writing to file for band plotting
    if (kept < nband)
      nband = kept;
    end
    for i = 1:nband
      fprintf(f2,"%15.6G",E(i));
    end
    fprintf(f2, "\n");
  end
end

fclose(f2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Step 9: Calculating and Plotting DOS


fprintf("\nNow calculating density of states. Please wait..")

datafile= [MaterialName,"-EigenMatrix.dat"]; % Reading data file for specific material

specify_format= 'yes'; % If yes, the two next definitions will be used
delim_in= ' ';         % Expected column separator character
head_in =  1;          % Expected number of lines of header

if(strcmp(specify_format,'yes'))
  [z,delim_out,head_out]=importdata(datafile,delim_in,head_in);
else
  [z,delim_out,head_out]=importdata(datafile);
end

if(head_out>0)
  x=z.data; % If header found, file content read as "structure array"
            % (Display z to see what is meant)
else
  x=z;      % If header not found, file content read as numerical array
end
MyEnergyData=[];
for i=1:nband
  MyEnergyData=[MyEnergyData;x(:,i)];
end
sorted=sort(MyEnergyData);
ENERGY = [-14:0.05:6]';
DOS=zeros(1,length(ENERGY))';

for i = 1:length(ENERGY)
      ExpoMatrix = (exp(-((ENERGY(i,1)-sorted).^2)/Smear^2))/(Smear*sqrt(pi));
      DOS(i,1) = sum(ExpoMatrix(:));
      DOS(i,2) = ENERGY(i,1);
endfor
DFN=['E',num2str(Setpot),'-',MaterialName,'-','MQ_r',num2str(MQ_r),'-','Smear',num2str(Smear),'-','DOSdata.dat']
f3 = fopen(DFN,'w');
for i=1:length(DOS)
  fprintf(f3,"%15.6G %15.6G\n",DOS(i,1),DOS(i,2));
end
fclose(f3)
source('DOS_Plot.m')
```

F. Code for Density of States (DOS) Plotting

```
datafile= ['E',num2str(Setpot),'-',MaterialName,'-','MQ_r',num2str(MQ_r),'-','Smear',num2str(Smear),'-','
    DOSdata.dat'];

% Reading data file for specific material

specify_format= 'yes'; % If yes, the two next definitions will be used
delim_in= ' ';         % Expected column separator character
head_in =  1;          % Expected number of lines of header

labelx = 'Wavevector [2\pi/a units]'; % NB: Greek characters in TeX encoding
labely = 'Energy [eV]';

if(strcmp(specify_format,'yes'))
  [z,delim_out,head_out]=importdata(datafile,delim_in,head_in);
else
  [z,delim_out,head_out]=importdata(datafile);
end

fprintf('\nFOUND IN DATA FILE:\n');
fprintf('Column separator character = ''%s''\n',delim_out);
fprintf('Number of lines of header  = %d\n',head_out);
if(head_out>0)
  x=z.data; % If header found, file content read as "structure array"
            % (Display z to see what is meant)
else
  x=z;      % If header not found, file content read as numerical array
end

sz=size(x);
fprintf('abscissae = %d\n',sz(1))
fprintf('columns   = %d\n',sz(2))
x(sz(1,1),1)=0;
x(1,1)=0;
for i=1:length(x(:,1))
  if x(i,2)<0
    x1(i,1)=x(i,1);
    x1(i,2)=x(i,2);
  else
    x2(i,1)=x(i,1);
    x2(i,2)=x(i,2);
  endif
endfor
MaximumDensity=[max(x1(:,1)),max(x2(:,1))];
MaxDens=max(MaximumDensity);

myplot = figure()
##plot(x1(:,2),x1(:,1)); % Remember: Sourced plot defaults will apply!
h = area(x1(:,1)/MaxDens, x1(:,2));
set(h, 'FaceColor',[0.5 0.5 0.5]);
hold on
h1 = area(x2(:,1)/MaxDens, x2(:,2));
set(h1, 'FaceColor', 'w');
hold on
line(xlim,[0,0],"linestyle", "-.",'color','k','linewidth',0.2)
legend('Filled Bands','Empty Bands', "location", "southeast")
hold on


hold on;
ylim([-14,6]);

set(gca,'xtick',[]);
set(gca, 'ytick', [-14:2:6])
set(legend,'FontSize',6)
## mytitle=["Band Structure of ",MaterialName];
## title(mytitle,"interpreter","latex")
xlabel("Density of States (DOS) $[eV^{-1}]$","interpreter","latex")
ylabel("Energy (eV)","interpreter","latex")


        %           Next lines must come after calling plot()


```

```
73         % OUTPUT FILE NAME
74
75 ## dot=rindex(datafile, '.');      % Position of filetype extension delimiter
76 ## root=substr(datafile,1,dot-1);  % Filename without filetype extension
77 ## outfile=sprintf('%s.pdf',root);
78 ##
79 ##         %OUTPUT FILES
80 ## print(outfile,'-dpdf');                % Basic pdf output
81 ## print(outfile,'-dpdflatexstandalone'); % Combined pdf & LaTeX files
82 ##
83 ## fprintf('\nOUTPUT FILES: \n');
84 ## fprintf('%s (basic pdf file)\n',outfile);
85 ## fprintf('%s.tex and %s-inc.pdf (for perfect LaTeX processing)\n',root,root);
86
87  waitfor(myplot); % Wait for closing graphical window
```