Outline of tb.c Programming

Required Utility Functions: https://aiichironakano.github.io/phys516/src/TB/

1. tb_util_angstrom.c or tb_util_au.c: Copy contents of either one into your header and program files (name them tb.h and tb.c, respectively)
2. eigen.c: To be compiled together with your tb.c

Additional Utility Function: https://aiichironakano.github.io/phys516/src/MD/

1. *SignR*() function in md.h to implement periodic boundary condition (PBC)

Algorithm

Read *InnitUcell*[3] // How many times to repeat the diamond unit cell (in tb_util.c)
InitConf() → $nAtom \leftarrow 8\prod_{a=0}^{2} IntiUcell[a]$
$r[0:nAtom-1][0:2]$ // $r[i][0|1|2] = r_{i,x|y|z}$, where $\vec{r}_i = (r_{i,x}, r_{i,y}, r_{i,z})$

int *n4* = 4×*nAtom* // Vector length
double** *h* = *dmatrix*(1,*n4*,1,*n4*) // Allocate memory and allow the use of *h*[1:*n4*][1:*n4*]
double* *d* = *dvector*(1,*n4*) // Allocate *d*[1:*n4*]
double* *e* = *dvector*(1,*n4*) // Allocate *e*[1:*n4*]; *dmatric*(0 and *dvector*() defined in eigen.c*

// Set up *h*[1:*n4*][1:*n4*]
for $1 \le i \le n4$
  for $1 \le j \le n4$
    $h[i][j] \leftarrow 0$
for $0 \le i < nAtom$
  // Fill diagonal (intra-atom) block
  $\begin{cases} h[4i+1][4i+1] \leftarrow E_s \\ h[4i+k][4i+k] \leftarrow E_p (k = 2,3,4) \end{cases}$
  for $i+1 \le j < nAtom$
    $\vec{r}_{ij} \leftarrow \vec{r}_i - \vec{r}_j = (r_{ij,x}, r_{ij,y}, r_{ij,z})$
    // Pick the minimum image (*cf. ComputeAccel*() in md.c)
    $r_{ij,a} = r_{ij,a} - SignR\left(\frac{L_a}{2}, r_{ij,a} - \frac{L_a}{2}\right) - SignR\left(\frac{L_a}{2}, r_{ij,a} + \frac{L_a}{2}\right)$ $(a = 0,1,2)$
    Here, $\frac{L_a}{2} = \frac{1}{2} \times \underbrace{LCNS}_{\text{lattice constant}} \times InitUcell[a] \rightarrow RegionH[3]$
    $\hat{d} \leftarrow \vec{r}_{ij} / \underbrace{|\vec{r}_{ij}|}_{r} = (d_x, d_y, d_z)$ // Unit-length bond-direction vector
    Fill off-diagonal block, $h[4i+k][4j+l]$ $(k,l = 1,2,3,4)$
    Symmetric copy: $h[4j+l][4i+k] = h[4i+k][4j+l]$ $(k,l = 1,2,3,4)$

* Don't forget to prototype *dmatrix*() and *dvector*(), which are defined in eigen.c, in your header file, tb.h.
  double **dmatrix(int, int, int, int);
  double *dvector(int, int);

// Diagonalize the Hamiltonian matrix[†]

tred2(*h*,*n4*,*d*,*e*)

tqli(*d*,e,*n4*,*h*) // Now *d*[1:*n4*] hold the eigenvalues (not sorted)

// Sort the eigenvalues in ascending order[‡]

for $1 \le i < n4$

  for $i + 1 \le j \le n4$

    if $d[i] > d[j]$

      // Swap the elements

      $dummy \leftarrow d[i]$

      $d[i] \leftarrow d[j]$

      $d[j] \leftarrow dummy$

// Plot the electronic density of states (DOS)

ε_{min} = -15.0 [eV] // Minimum and maximum energies to plot DOS

ε_{max} = 10.0 [eV]

σ = 0.1 [eV] // Smear each eigenvalue as a Gaussian function

for $0 \le i < Nbin$ // Use the number of energy bins *Nbin* = 500

  $\varepsilon_i \leftarrow \varepsilon_{min} + i\frac{\varepsilon_{max}-\varepsilon_{min}}{Nbin}$

  $D(\varepsilon_i) \leftarrow \sum_{v=1}^{n4} \frac{1}{\sqrt{\pi}\sigma} \exp\left[-\frac{(\varepsilon_i-\varepsilon_v)^2}{\sigma^2}\right]$

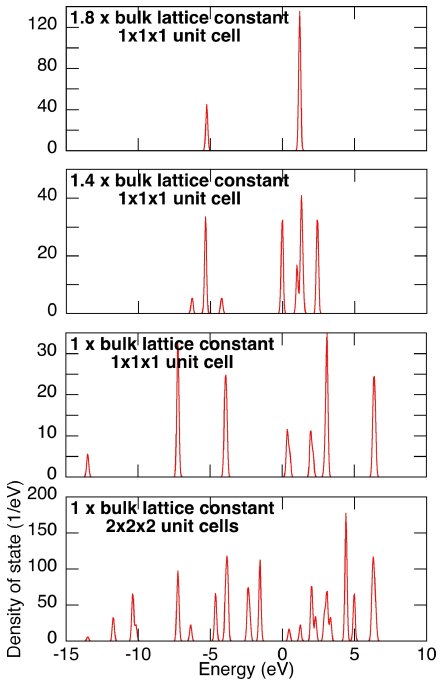
  Plot ε_i *vs.* $D(\varepsilon_i)$

Compile and Run

Compile: cc –o tb tb.c eigen.c -lm

Run: *LCNS* = 5.43 Å (or 10.2622 a.u.) × 1.0 | 1.4 | 1.8;
InitUcell[3] = (1, 1, 1), 8 atoms

LCNS = 5.43 Å (or 10.2622 a.u.);
InitUcell[3] = (2, 2, 2), 64 atoms



[†] Don't forget to prototype *tred2*() and *tqli*(), which are defined in eigen.c, in your header file, tb.h.
  void tred2(double **, int, double *, double *);
  void tqli(double *, double *, int, double **);

[‡] Upon return from *tqli*(), *h*[1:*n4*][*i*] and *h*[1:*n4*][*j*] store *i*-th and *j*-th eivenvectors; these columns need be swapped as well, if we need to use them (not necessary for the homework).