Lattice Simulation

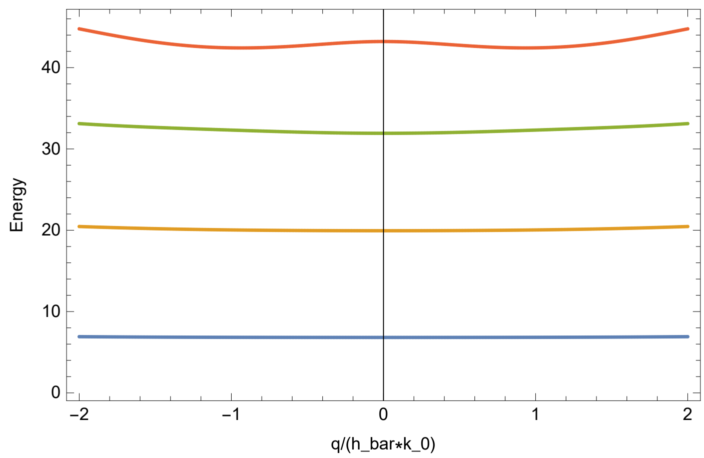
Bloch wave basis, periodic potential

Dispersion relation

In[165]:=

```
Hij[V0_, q_, i_, j_] := If[i == j, (2 i + q)^2 + V0 / 2, If[Abs[i - j] == 1, -1/4 V0, 0]];
(*Hamiltonian matrix elements*)
lMax = 3;  (*maximum absolute value of bloch wave index*)
H[V0_, q_] := Table[Hij[V0, q, i, j], {i, -lMax, lMax}, {j, -lMax, lMax}]
(* Hamiltonian matrix form *)
eigenvals[V_, q_] := Sort[N[Eigenvalues[H[V, q]]]]
(* Get eigenvalues of matrix (eigenenergies) *)
V0 = 50;
qlist = Range[-2, 2, 0.01]; (*quasi-momentum list from -q0 to q0*)
EnergyMx = Map[eigenvals[V0, #] &, qlist];
(* list of lists: energy eigenvalues for each quasi momentum value q *)
nband = 4;
(*Plot to nth band, which should be smaller than (2*lmax+1)*)
FigBS2 = ListPlot[Table[Transpose[{qlist, EnergyMx[[All, iband]]}], {iband, 1, nband}],
  Frame → True, FrameLabel → {"q/(h_bar*k_0)", "Energy"}](*listplot is fast*)
```

Out[168]=



Band Gaps

In[169]:=
```
BandGapMin = Table[Min[EnergyMx[[All, jband]] - EnergyMx[[All, 1]]], {jband, 2, nband}];
(*min band gap 1-2, 1-3, 1-4, ... till 1-nband *)
BandGapMax = Table[Max[EnergyMx[[All, jband]] - EnergyMx[[All, 1]]], {jband, 2, nband}];
(*max band gap ..*)
BandGapMean = Table[Mean[EnergyMx[[All, jband]] - EnergyMx[[All, 1]]],
   {jband, 2, nband}];(*mean band gap ..*)
BandGapTb = Transpose[{BandGapMin, BandGapMax, BandGapMean}];
FirstColumn = Prepend[Table["1-" <> ToString[j], {j, 2, nband}], "Band Gap"];
DataColumns = Prepend[BandGapTb, {"Min", "Max", "Mean"}];
BandGapTbLabel = MapThread[Prepend, {DataColumns, FirstColumn}];
Grid[BandGapTbLabel, Dividers → {False, All}]
```

Out[176]=

Band Gap	Min	Max	Mean
1-2	13.1237	13.5473	13.2495
1-3	25.1051	26.2017	25.5271
1-4	35.5884	37.8538	36.196

Tunneling

In[177]:=
```
BandMax = Map[Max[EnergyMx[[All, #]]] &, Range[1, nband]];
(* max energy for each band *)
BandMin = Map[Min[EnergyMx[[All, #]]] &, Range[1, nband]];
(* min energy for each band *)
BandWidth = BandMax - BandMin;(* band width for each band *)
Tunneling = BandWidth / 4; (* saw in Dieter Jaksch PRL paper 1998,
which is valid both for all bands if the band is deep enough such as 5Er*)
FirstColumn = Prepend[Table["Band" <> ToString[j], {j, 1, nband}], " "];
DataColumns = Prepend[Tunneling, "Tunneling"];
Grid[Transpose[{FirstColumn, DataColumns}], Dividers → {False, All}]
```

Out[183]=

	Tunneling
Band1	0.0237072
Band2	0.129613
Band3	0.29787
Band4	0.586689

In[148]:=

" "	"Tunneling"
"Band1"	0.019192806433558474`
"Band2"	0.2500458969192565`
"Band3"	0.8936185520855338`
"Band4"	1.6490815341159566`

Out[148]=

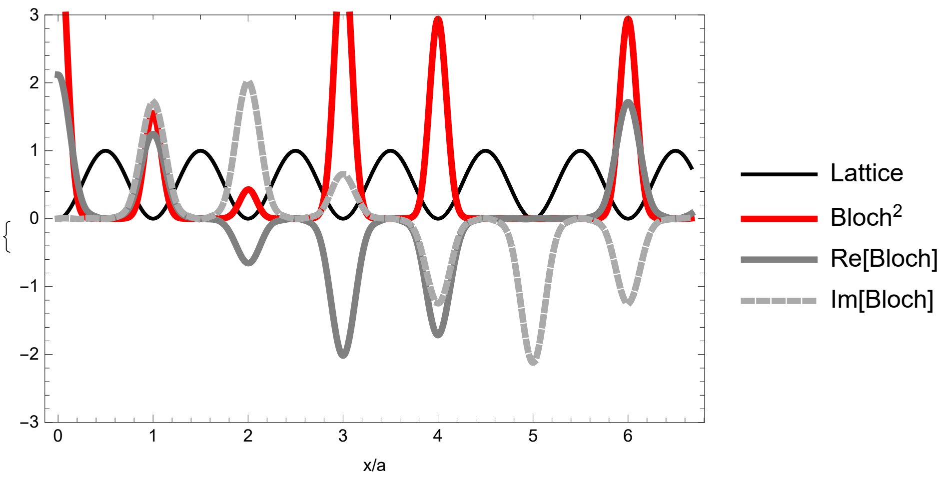
	Tunneling
Band1	0.0191928
Band2	0.250046
Band3	0.893619
Band4	1.64908

Bloch Waves

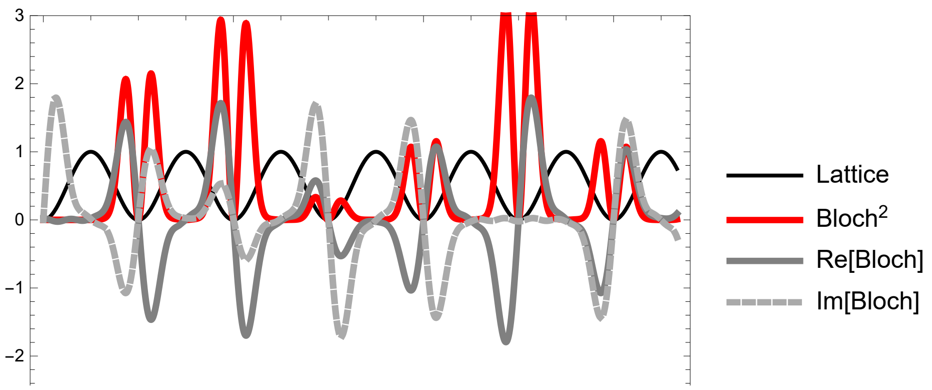
In[184]:=
```
EV[V_, q_, l_] := Sort[Transpose[Chop[N[Eigensystem[H[V, q]]]]]] [[l, 2]];
(* Get the l th eigen vector of H , there are 2*lmax + 1 in total*)
EVNormed[V_, q_, l_] := Module[{tmp = EV[V, q, l]}, (tmp / Norm[tmp])] ;
(* Get normed l th eigen vector of H,
gives 2*lmax+1 probabilities of different Bloch wave states *)
Bloch[x_, V_, q_, l_] :=
  Sum[Exp[π * I (q + 2 n) x] * EVNormed[V, q, l] [[n + lMax + 1]], {n, -lMax, lMax}];
(* Bloch wave is summation of plane waves *)
(*should we use cutoff jMax or smaller one instead??*)
q = 0.3
oneoverq = If[q > 0, 1 / q, 3];
BRe[x_] = {Re[ Bloch[x, V0, q, 1]],
   Re[ Bloch[x, V0, q, 2]], Re[ Bloch[x, V0, q, 3]], Re[ Bloch[x, V0, q, 4]]};
BIm[x_] = {Im[ Bloch[x, V0, q, 1]], Im[ Bloch[x, V0, q, 2]],
   Im[ Bloch[x, V0, q, 3]], Im[ Bloch[x, V0, q, 4]]};
BNorm[x_] = {Re[ Bloch[x, V0, q, 1]], Re[I Bloch[x, V0, q, 2]],
   Re[Bloch[x, V0, q, 3]], Re[ Bloch[x, V0, q, 4]]};
Table[Plot[{Sin[π * x]^2, BRe[x]^2[[i]], BRe[x] [[i]], BIm[x] [[i]]},
  {x, 0, 2 * oneoverq}, Frame → True, Axes → False, FrameLabel → {"x/a"},
  PlotRange → {-3, 3}, PlotLegends → {"Lattice", "Bloch^2", "Re[Bloch]", "Im[Bloch]"},
  PlotStyle → {{Thick, Black}, {Thickness[0.01], Red}, {Thickness[0.01], Gray},
    {Thickness[0.01], Dashed, Lighter[Gray]}}, ImageSize → Medium], {i, 1, 4}]
```

Out[187]= 0.3

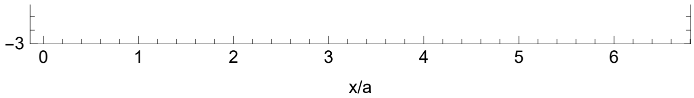
Out[192]= {

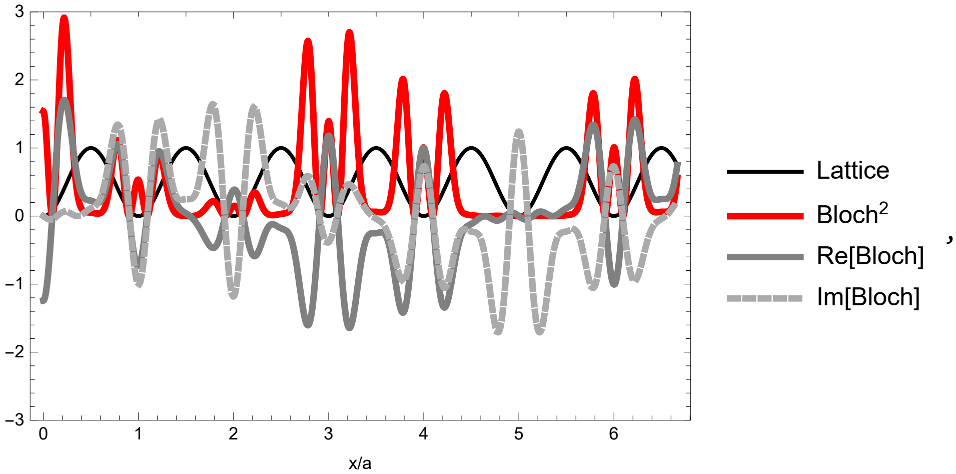


,

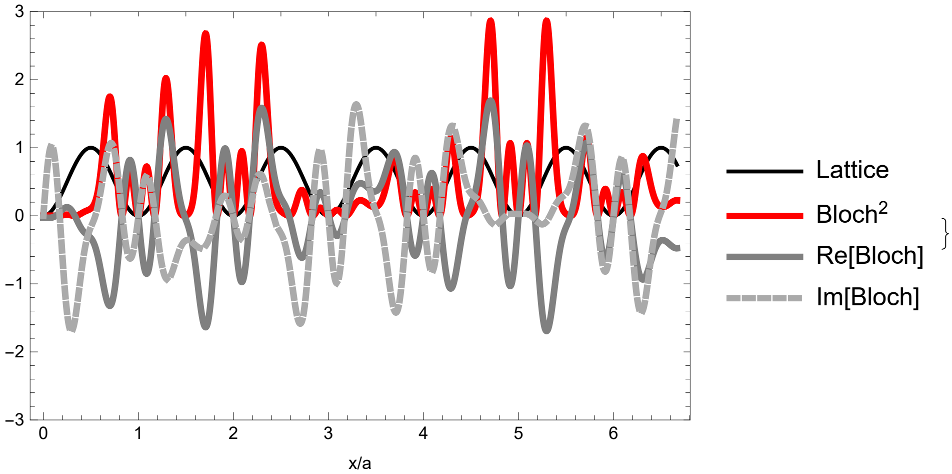


,





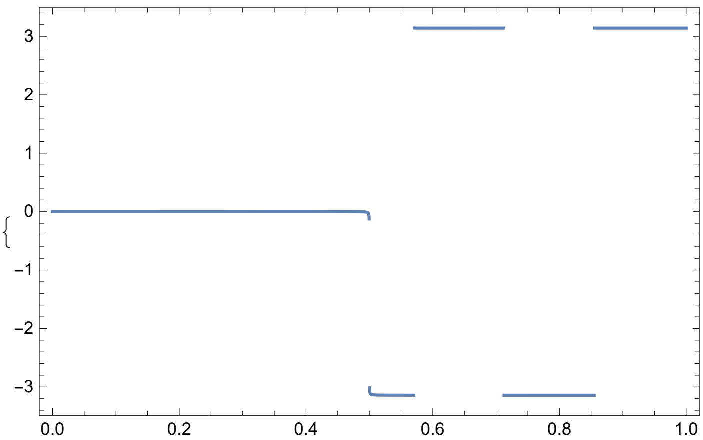
,



}

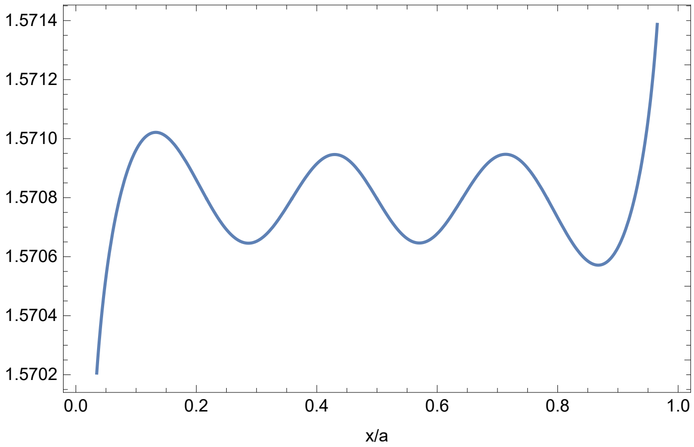
In[87]:=
```
q = 1;
oneoverq = If[q > 0, 1/q, 3];
BArg[x_] = {Arg[Bloch[x, V0, q, 1]],
   Arg[Bloch[x, V0, q, 2]], Arg[Bloch[x, V0, q, 3]], Arg[Bloch[x, V0, q, 4]]};
Table[Plot[Chop[BArg[x][[i]]], {x, 0, oneoverq}, Frame → True,
  Axes → False, FrameLabel → {"x/a"}, ImageSize → Medium], {i, 1, 4}]
Arg[Bloch[0, V0, 1, 2]]
```

Out[90]= {

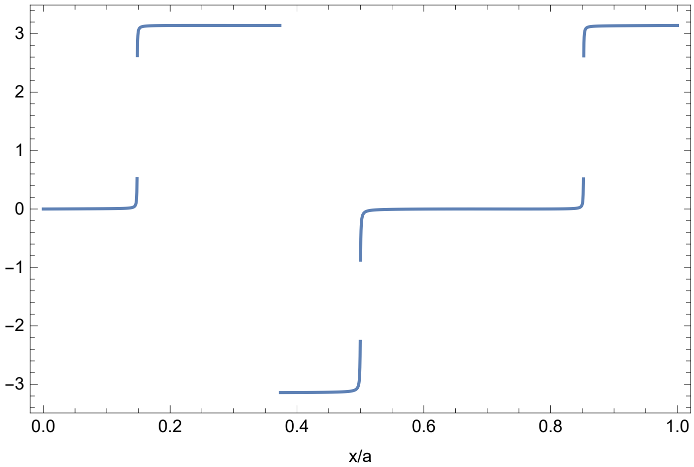


,

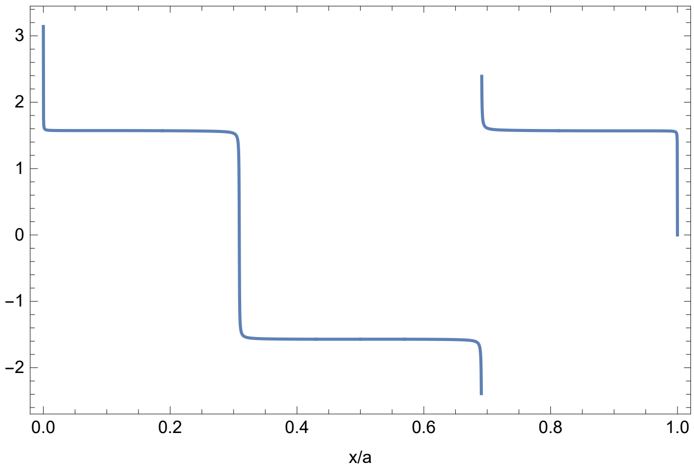
x/a


1.5714
1.5712
1.5710
1.5708
1.5706
1.5704
1.5702
0.0
0.2
0.4
0.6
0.8
1.0
x/a

,


3
2
1
0
−1
−2
−3
0.0
0.2
0.4
0.6
0.8
1.0
x/a

,


3
2
1
0
−1
−2
0.0
0.2
0.4
0.6
0.8
1.0
x/a

}

Out[91]= 0

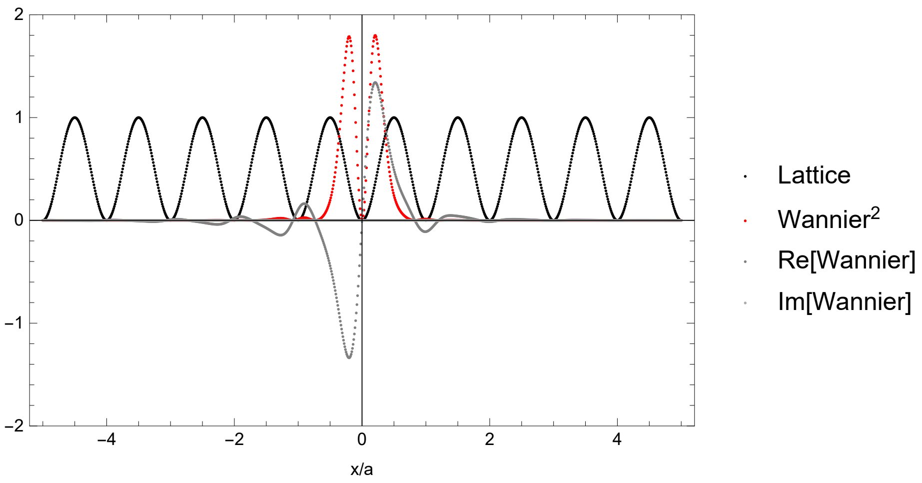
Wannier Functions

In[100]:=
```
PhaseFactor[V_, q_, l_] :=
  If[OddQ[l], Arg[Bloch[0, V, q, l]], Arg[Bloch[0.25, V, q, l]]];
(* for odd bands, take phase factor at x = 0. If l is even,
take phase factor at x = 0.25 *)
(*force phase at x=0 equals to 0 for every Bloch waves for odd band and phase at x=
  0.25 (or values close to 0.13) equals to 0 for even band. It just Works!!*)
qqstep = 0.1;
q = 0;
Table[q, {-1, 1, qqstep}]
Wannier[x_, V_, l_, xi_] :=
 Sum[Exp[π * I * q * (-xi)] * Exp[-I * PhaseFactor[V, q, l]] * Bloch[x, V, q, l],
    {q, -1, 1, qqstep}]  *  (qqstep / 2)  -
  1 / 2 (Sum[Exp[π * I * q * (-xi)] * Exp[-I * PhaseFactor[V, q, l]] * Bloch[x, V, q, l],
     {q, -1, 1, 2}]) * (qqstep / 2)
(*numerically integrate q from -1 to 1, however -
 1 and 1 are the same and we should deduct one*)
(*x: position, V: lattice depth, l: nth band,
xi: wannier function at lattice site xi, which should be an integer*)
(*Wannier[x_,V_,l_,xi_]:=
 Sum[Exp[π*I*q*(-xi)]*SignFactor[V,q,l]*Bloch[x,V,q,l],{q,-1,1,qqstep}]*(qqstep/2)-
  1/2(Sum[Exp[π*I*q*(-xi)]*SignFactor[V,q,l]*Bloch[x,V,q,l],{q,-1,1,2}])*
   (qqstep/2)*)
nWannier = 2; (*1st, 2nd, 3rd band*)
xxstep = 0.01; xlist = Range[-5, 5, xxstep];
wlist = Wannier[xlist, Vlat, nWannier, 0];
ltslist = Sin[π * xlist]^2; (*lattice potential*)
wlistNorm = wlist / Sqrt[Total[Re[wlist]^2 * xxstep]];
(*Normalize the wannier function. Here wannier function is assumed to be real*)
ListPlot[{Transpose[{xlist, ltslist}],
  Transpose[{xlist, Re[wlistNorm]^2}], Transpose[{xlist, Re[wlistNorm]}],
  Transpose[{xlist, Im[wlistNorm]}]}, FrameLabel → {"x/a"}, Frame → True,
 PlotLegends → {"Lattice", "Wannier^2", "Re[Wannier]", "Im[Wannier]"},
 PlotRange → {-2, 2}, PlotStyle → {{Thick, Black}, {Thickness[0.01], Red},
   {Thickness[0.01], Gray}, {Thickness[0.01], Dashed, Lighter[Gray]}}]
```

... Table: Raw object -1 cannot be used as an iterator.

Out[103]=
```
Table[q, {-1, 1, qqstep}]
```

Out[107]=



Bloch wave basis, periodic potential + linear term

Dispersion relation