By hand calculation of band structure, TDOS, and PDOS

It is assumed that you have done the crystal field calculation and the Monte Carlo calculation at this point.

Turn off errors

Crystal Field Hamiltonian

Crystal field Hamiltonian defintion

```
In[3]:= (*Cutoff parameter *)
        t = 0.2;

        (*Functions for a Cartesian point given spherical coordinates*)
        x1[rho_, theta_, phi_] := rho*Sin[theta]*Cos[phi]
        y1[rho_, theta_, phi_] := rho*Sin[theta]*Sin[phi]
        z1[rho_, theta_, phi_] := rho*Cos[theta]
        P[rho_, theta_, phi_] :=
          {x1[rho, theta, phi], y1[rho, theta, phi], z1[rho, theta, phi]}

        (*Pyramid  and tetrahedron vertices*)
        pyr =
            {{2.51559, 0., 0.}, {-0.518164, -1.75154, -1.8285}, {-0.518164, -1.75154, 1.8285},
              {-0.518164, 1.75154, -1.8285}, {-0.518164, 1.75154, 1.8285}};
        tetra = {{-1.5077737499999997, 0., -1.8285}, {-1.5077737499999997, 0., 1.8285},
              {1.5259762500000003, -1.751536379154027, 0.},
              {1.5259762500000003, 1.751536379154027, 0.}};

        (*Crystal field Hamiltonian *)
        H1Cr[rho_, theta_, phi_] :=
          Sum [(1/Sqrt[((Norm [P[rho, theta, phi] - pyr[[i]]]) ^2 + t ^2]), {i, 1, Length@pyr}]
        H1Fe[rho_, theta_, phi_] := Sum [
            (1/Sqrt[((Norm [P[rho, theta, phi] - tetra[[i]]]) ^2 + t ^2]), {i, 1, Length@tetra}]
        H1Cart[x_, y_, z_] := Sum [(1/Sqrt[((Norm [{x, y, z} - pyr[[i]]]) ^2 + t ^2]),
            {i, 1, Length@pyr}]
```

Definition of Cartesian d orbital states for Cr and Fe

In[13]:=

```
a0 = 1;
```

$$\rho[n\_,\, r\_] := \frac{2\,r}{n\,a0};$$

$$\psi\text{nlm}\;[n\_,\, l\_,\, m\_\;,\, r\_,\, \theta\_,\, \phi\_] := \sqrt{\left(\frac{2}{n\,a0}\right)^{3} \frac{(n-l-1)\,!}{2\,n\,(n+l)\,!}}\; e^{-\rho[n,r]/2}\, \rho[n,\, r]^{l}$$
$$\quad \text{LaguerreL}[n-l-1,\, 2\,l+1,\, \rho[n,\, r]]\; \text{SphericalHarmonicY}\;[l,\, m\;,\, \theta,\, \phi]$$

$$\psi\text{nlmCart}\;[n\_,\, l\_,\, m\_\;,\, x\_,\, y\_,\, z\_] := \sqrt{\left(\frac{2}{n\,a0}\right)^{3} \frac{(n-l-1)\,!}{2\,n\,(n+l)\,!}}\; e^{-\rho[n,r]/2}\, \rho[n,\, r]^{l}$$
$$\quad \text{LaguerreL}[n-l-1,\, 2\,l+1,\, \rho[n,\, r]]\; \text{SphericalHarmonicY}\;[l,\, m\;,\, \theta,\, \phi]\; /.$$
$$\quad \left\{r \to \sqrt{x^2+y^2+z^2}\;,\, \theta \to \text{ArcCos}\left[\frac{z}{\sqrt{x^2+y^2+z^2}}\right],\, \phi \to \text{ArcCos}\left[\frac{x}{\sqrt{x^2+y^2}}\right]\right\}$$

```
ψ4sNorm [x_, y_, z_] := ψnlm [1, 0, 0, r, θ, ϕ] /. r → Sqrt[x^2 + y^2 + z^2]
R32[r_] := r^2 E^(-r / (3 a0));
ψxy[x_, y_, z_] := (x y) / r^2 R32[r] /. r → Sqrt[x^2 + y^2 + z^2]
Nxy = 1/(√NIntegrate[ψxy[x, y, z]*ψxy[x, y, z], {x, -∞, ∞}, {y, -∞, ∞}, {z, -∞, ∞}]);
ψxyNorm [x_, y_, z_] := Nxy ψxy[x, y, z]
ψxz[x_, y_, z_] := (x z) / r^2 R32[r] /. r → Sqrt[x^2 + y^2 + z^2]
Nxz = 1/(√NIntegrate[ψxz[x, y, z]*ψxz[x, y, z], {x, -∞, ∞}, {y, -∞, ∞}, {z, -∞, ∞}]);
ψxzNorm [x_, y_, z_] := Nxz ψxz[x, y, z]
ψyz[x_, y_, z_] := (y z) / r^2 R32[r] /. r → Sqrt[x^2 + y^2 + z^2]
Nyz = 1/(√NIntegrate[ψyz[x, y, z]*ψyz[x, y, z], {x, -∞, ∞}, {y, -∞, ∞}, {z, -∞, ∞}]);
ψyzNorm [x_, y_, z_] := Nyz ψyz[x, y, z]
ψz2[x_, y_, z_] := (3 z^2 - r^2) / r^2 R32[r] /. r → Sqrt[x^2 + y^2 + z^2]
Nz2 = 1/(√NIntegrate[ψz2[x, y, z]*ψz2[x, y, z], {x, -∞, ∞}, {y, -∞, ∞}, {z, -∞, ∞}]);
ψz2Norm [x_, y_, z_] := Nz2 ψz2[x, y, z]
ψx2y2[x_, y_, z_] := (x^2 - y^2) / r^2 R32[r] /. r → Sqrt[x^2 + y^2 + z^2]
Nx2y2 =
    1/(√NIntegrate[ψx2y2[x, y, z]*ψx2y2[x, y, z], {x, -∞, ∞}, {y, -∞, ∞}, {z, -∞, ∞}]);
ψx2y2Norm [x_, y_, z_] := Nx2y2 ψx2y2[x, y, z]

ψCentral[x_, y_, z_] := {ψ4sNorm [x, y, z], ψxyNorm [x, y, z],
    ψxzNorm [x, y, z], ψyzNorm [x, y, z], ψz2Norm [x, y, z], ψx2y2Norm [x, y, z]}
pyrState[x_, y_, z_] := ψCentral[x, y, z][[2 ;; 6]];
pyrStateb[rho_, theta_, phi_] := pyrState[x, y, z] /.
    {x → rho Sin[theta] Cos[phi], y → rho Sin[theta] Sin[phi], z → rho Cos[theta]}
```

Express crystal field Hamiltonian in basis of Cartesian d orbital states

```
In[37]:= range = ∞;
         (*Matrix Elements  for Crystal Field Hamiltonian *)
         VCr[i_, j_] := NIntegrate[pyrStateb[rho, theta, phi][[i]] *
              H1Cr[rho, theta, phi] * pyrStateb[rho, theta, phi][[j]] rho^2 Sin[theta],
            {rho, 0, range}, {theta, 0, π}, {phi, 0, 2 π}]
         VFe[i_, j_] := NIntegrate[pyrStateb[rho, theta, phi][[i]] *
              H1Fe[rho, theta, phi] * pyrStateb[rho, theta, phi][[j]] rho^2 Sin[theta],
            {rho, 0, range}, {theta, 0, π}, {phi, 0, 2 π}]
```

Calculate all matrix elements and put them in a table

```
In[43]:= HCFCr = Monitor[Table[VCr[i, j], {i, 5}, {j, 5}], {i, j}];
         HCFFe = Monitor[Table[VFe[i, j], {i, 5}, {j, 5}], {i, j}];
         eigsCr= Chop@Eigensystem @HCFCr;
         eigsFe= Chop@Eigensystem @HCFFe;
```

Output the results of crystal field splitting

```
In[47]:= eigsCr[[2]] >> "~/pyramid /CFSResults/eigsCr.txt";
         eigsFe[[2]] >> "~/tetrahedron/CFSResults/eigsFe.txt";
```

Definition of unitary matrices from MC

```
In[40]:= unitary[overlapMatrix_, m_  , n_, θ_] :=
           (
             unitaryMatrix= IdentityMatrix@Length@overlapMatrix;
             unitaryMatrix[[m  , m  ]] = Cos [θ];
             unitaryMatrix[[m  , n]] = Sin[θ];
             unitaryMatrix[[n, m  ]] = -Sin[θ];
             unitaryMatrix[[n, n]] = Cos [θ];
             unitaryMatrix
           )

         unitaryMatrices[overlapMatrix_, changesArrayVar_] :=
             unitary[overlapMatrix, changesArrayVar[[#, 1, 1]], changesArrayVar[[#, 1, 2]],
                  changesArrayVar[[#, 1, 3]]] & /@Range@Length@changesArrayVar;

         construct𝒰tot[overlapMatrix_, changesArrayVar_] :=
           (
             unitaryMatricesList=
               unitary[overlapMatrix, changesArrayVar[[#, 1, 1]], changesArrayVar[[#, 1, 2]],
                    changesArrayVar[[#, 1, 3]]] & /@Range@Length@changesArrayVar;
             𝒰tot = IdentityMatrix[Length@overlapMatrix];
             Do[𝒰tot = unitaryMatricesList[[i]].𝒰tot;
               , {i, Length@unitaryMatricesList}];
             𝒰tot
           )
```

Import results from file to define crystal field effect in Hamiltonian

```
In[117]:= CFS𝒰Cr = << "~/pyramid /CFSResults/eigsCr.txt";
          CFS𝒰Fe = << "~/tetrahedron/CFSResults/eigsFe.txt";
          pyrData = << "~/pyramid /ChangesArray/changesArray_pyramid .txt";
          tetraData = << "~/tetrahedron/ChangesArray/changesArray_tetrahedron.txt";
          pyr𝒰 = construct𝒰tot[IdentityMatrix[5], pyrData];
          tetra𝒰 = construct𝒰tot[IdentityMatrix@5, tetraData].
                 {{1, 0, 0, 0, 0}, {0, 1, 0, 0, 0}, {0, 0, 0, 0, 1}, {0, 0, 0, 1, 0}, {0, 0, 1, 0, 0}};
          𝒰Cr = CFS𝒰Cr.Transpose@pyr𝒰;
          𝒰Fe = CFS𝒰Fe.Transpose@tetra𝒰;

          eigvals= << "~/pyramid /CFSResults/eigvalsCr.txt";
          eigvals2= << "~/tetrahedron/CFSResults/eigvalsFe.txt";

          offDiagonal[matrixTest_ , unitary𝒰_] :=
            Chop[unitary𝒰†.DiagonalMatrix[matrixTest ].unitary𝒰]

          defineCrystalFields[splitCr_, splitFe_] :=
             (
               splitMatCr= offDiagonal[splitCr (eigvals- Mean@eigvals), 𝒰Cr];
               𝒰2Cr =
                 {{0, 0, 0, 0, 1}, {0, 0, 0, 1, 0}, {0, 0, 1, 0, 0}, {0, 1, 0, 0, 0}, {1, 0, 0, 0, 0}};
               CFSCr = 𝒰2Cr†.splitMatCr.𝒰2Cr;

               splitMatFe= offDiagonal[splitFe (eigvals2- Mean@eigvals2), 𝒰Fe];
               𝒰2Fe =
                 {{0, 1, 0, 0, 0}, {1, 0, 0, 0, 0}, {0, 0, 0, 1, 0}, {0, 0, 1, 0, 0}, {0, 0, 0, 0, 1}};
               CFSFe = 𝒰2Fe†.splitMatFe.𝒰2Fe;
             )

In[129]:= defineCrystalFields[1000, 1000]
```

Define primitive vectors and reciprocal vectors for k path definition

In[61]:=
```
a = 6.0675;
c = 3.6570;
primvecs = {a {1, 0, 0}, a {-1/2, Sqrt[3]/2, 0}, c {0, 0, 1}};
origin = {0, 0, 0};
reciprocalVectors[primitiveVectors_] :=
  (
    a1 = primitiveVectors[[1]];
    a2 = primitiveVectors[[2]];
    a3 = primitiveVectors[[3]];
    2 π {Cross[a2, a3]/(a1.Cross[a2, a3]), Cross[a3, a1]/(a2.Cross[a3, a1]), Cross[a1, a2]/(a3.Cross[a1, a2])}
  )
recVectors = reciprocalVectors@primvecs ;

kvec[kx_, ky_, kz_] := {kx, ky, kz};

point1 = .5 recVectors[[1]];
point2 = {.25 Norm @recVectors[[1]], .5 Norm @recVectors[[1]], 0};
nPoints = 100;

kPath[t_] := Piecewise[{{point1 t, 0 ≤ t ≤ 1}, {(point2 - point1) (t - 1) + point1, 1 ≤ t ≤ 2},
      {(origin - point2) (t - 2) + (point2), 2 ≤ t ≤ 3},
      {π/(2 c) {0, 0, 1} (t - 3), 3 ≤ t ≤ 4}, {point1 (t - 4) + π/(2 c) {0, 0, 1}, 4 ≤ t ≤ 5},
      {(point2 - point1) (t - 5) + point1 + π/(2 c) {0, 0, 1}, 5 ≤ t ≤ 6},
      {(origin - point2) (t - 6) + (point2) + π/(2 c) {0, 0, 1}, 6 ≤ t ≤ 7}}]
kPoints = kPath[#] & /@ Range[0, 7, 1/nPoints];
```

Definition of sectors and total Hamiltonian

Angles

```
In[73]:= angle1 = (π / 180) (180 - 90.0604);
         angle2 = (π / 180) (180 - 75.4757);
         angle3 = (π / 180) (180 - 138.6132);
         angle4 = (π / 180) (180 - 120);
         angle5 = (π / 180) (180 - 100.9823);
         angle6 = (π / 180) (180 - 142.9040);
         ang1 = (π / 180) (180 - 67.7109);
         ang2 = (π / 180) (180 - 70.925);
         ang3 = (π / 180) (180 - 134.9172);
         ang4 = (π / 180) (180 - 71.452);
         ang5 = (π / 180) (180 - 138.6132);
```

Chromium subsector and full sector

```
In[84]:= hamiltonian1[kx_, ky_, kz_, tz_] := tz Cos[angle4]
           {{0, e^(-i kvec[kx,ky,kz].(primvecs[[1]]+primvecs[[2]])), e^(-i kvec[kx,ky,kz].primvecs[[1]])},
            {e^(i kvec[kx,ky,kz].(primvecs[[1]]+primvecs[[2]])), 0, e^(i kvec[kx,ky,kz].primvecs[[2]])},
            {e^(i kvec[kx,ky,kz].primvecs[[1]]), e^(-i kvec[kx,ky,kz].primvecs[[2]]), 0}};

         hamiltonian2[kx_, ky_, kz_, tPerp_, tzp_, t2p_] :=
          {{0, tPerp Cos[angle2], tPerp Cos[angle2], tzp Cos[angle1] e^(i kvec[kx,ky,kz].primvecs[[3]]),
             t2p Cos[angle3] e^(i kvec[kx,ky,kz].primvecs[[3]]),
             t2p Cos[angle3] e^(i kvec[kx,ky,kz].primvecs[[3]])}, {tPerp Cos[angle2],
             0, tPerp Cos[angle2], t2p Cos[angle3] e^(i kvec[kx,ky,kz].primvecs[[3]]),
             tzp Cos[angle1] e^(i kvec[kx,ky,kz].primvecs[[3]]),
             t2p Cos[angle3] e^(i kvec[kx,ky,kz].primvecs[[3]])}, {tPerp Cos[angle2],
             tPerp Cos[angle2], 0, t2p Cos[angle3] e^(i kvec[kx,ky,kz].primvecs[[3]]),
             t2p Cos[angle3] e^(i kvec[kx,ky,kz].primvecs[[3]]),
             tzp Cos[angle1] e^(i kvec[kx,ky,kz].primvecs[[3]])},
           {tzp Cos[angle1] e^(-i kvec[kx,ky,kz].primvecs[[3]]),
             t2p Cos[angle3] e^(-i kvec[kx,ky,kz].primvecs[[3]]),
             t2p Cos[angle3] e^(-i kvec[kx,ky,kz].primvecs[[3]]), 0, tPerp Cos[angle2],
             tPerp Cos[angle2]}, {t2p Cos[angle3] e^(-i kvec[kx,ky,kz].primvecs[[3]]),
             tzp Cos[angle1] e^(-i kvec[kx,ky,kz].primvecs[[3]]),
             t2p Cos[angle3] e^(-i kvec[kx,ky,kz].primvecs[[3]]), tPerp Cos[angle2], 0,
             tPerp Cos[angle2]}, {t2p Cos[angle3] e^(-i kvec[kx,ky,kz].primvecs[[3]]),
             t2p Cos[angle3] e^(-i kvec[kx,ky,kz].primvecs[[3]]), tzp Cos[angle1]
               e^(-i kvec[kx,ky,kz].primvecs[[3]]), tPerp Cos[angle2], tPerp Cos[angle2], 0}};

         hamiltonian3[kx_, ky_, kz_, tPerp_, tzp_, t2p_] :=
          {{0, tPerp Cos[angle2] e^(-i kvec[kx,ky,kz].primvecs[[1]]), tPerp Cos[angle2]
                e^(i kvec[kx,ky,kz].primvecs[[2]]), tzp Cos[angle1] e^(i kvec[kx,ky,kz].primvecs[[3]]),
             t2p Cos[angle3] e^(i kvec[kx,ky,kz].(-primvecs[[1]]+primvecs[[3]])),
             t2p Cos[angle3] e^(i kvec[kx,ky,kz].(primvecs[[2]]+primvecs[[3]]))},
           {tPerp Cos[angle2] e^(i kvec[kx,ky,kz].primvecs[[1]]), 0,
        tPerp Cos[angle2] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[2]])),
        t2p Cos[angle3] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[3]])),
        tzp Cos[angle1] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]]),
        t2p Cos[angle3] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[2]]+primvecs [[3]]))},
      {tPerp Cos[angle2] ⅇ^(-ⅈ kvec[kx,ky,kz].primvecs [[2]]),
        tPerp Cos[angle2] ⅇ^(-ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[2]])),
        0, t2p Cos[angle3] ⅇ^(ⅈ kvec[kx,ky,kz].(-primvecs [[2]]+primvecs [[3]])),
        t2p Cos[angle3] ⅇ^(ⅈ kvec[kx,ky,kz].(-primvecs [[1]]-primvecs [[2]]+primvecs [[3]])),
        tzp Cos[angle1] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]])},
      {tzp Cos[angle1] ⅇ^(-ⅈ kvec[kx,ky,kz].primvecs [[3]]),
        t2p Cos[angle3] ⅇ^(-ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[3]])),
        t2p Cos[angle3] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[2]]-primvecs [[3]])),
        0, tPerp Cos[angle2] ⅇ^(-ⅈ kvec[kx,ky,kz].primvecs [[1]]),
        tPerp Cos[angle2] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[2]])},
      {t2p Cos[angle3] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]-primvecs [[3]])),
        tzp Cos[angle1] ⅇ^(-ⅈ kvec[kx,ky,kz].primvecs [[3]]),
        t2p Cos[angle3] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[2]]-primvecs [[3]])),
        tPerp Cos[angle2] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[1]]), 0,
        tPerp Cos[angle2] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[2]]))},
      {t2p Cos[angle3] ⅇ^(-ⅈ kvec[kx,ky,kz].(primvecs [[2]]+primvecs [[3]])),
        t2p Cos[angle3] ⅇ^(-ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[2]]+primvecs [[3]])),
        tzp Cos[angle1] ⅇ^(-ⅈ kvec[kx,ky,kz].primvecs [[3]]),
        tPerp Cos[angle2] ⅇ^(-ⅈ kvec[kx,ky,kz].primvecs [[2]]),
        tPerp Cos[angle2] ⅇ^(-ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[2]])), 0}};

hamiltonianCrCr[kx_, ky_, kz_, tz_, tPerp_, tzp_, t2p_, CFS_] :=
  ArrayFlatten[{{hamiltonian2[kx, ky, kz, tPerp, tzp, t2p], ConstantArray[0, {6, 6}],
        ConstantArray[0, {6, 3}]}, {ConstantArray[0, {6, 6}],
        hamiltonian3[kx, ky, kz, tPerp, tzp, t2p], ConstantArray[0, {6, 3}]},
      {ConstantArray[0, {3, 6}], ConstantArray[0, {3, 6}],
        hamiltonian1[kx, ky, kz, tz]}}] +
    ArrayFlatten[Table[IdentityMatrix[3] CFS[[i, j]], {i, 5}, {j, 5}]]
```

Iron sector

```
In[88]:= hamiltonianFeFeA [kx_, ky_, kz_, tzFe_, tzFep_, tperpFe_] := ArrayFlatten@
     {{ConstantArray[0, {3, 3}], {{tzFe Cos[angle5] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]]),
           tzFep Cos[angle6] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]]),
           tzFep Cos[angle6] ⅇ^(ⅈ kvec[kx,ky,kz].(-primvecs [[1]]-primvecs [[2]]+primvecs [[3]]))},
         {tzFep Cos[angle6] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]]),
           tzFe Cos[angle5] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]]),
           tzFep Cos[angle6] ⅇ^(ⅈ kvec[kx,ky,kz].(-primvecs [[1]]-primvecs [[2]]+primvecs [[3]]))},
         {tzFep Cos[angle6] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[2]]+primvecs [[3]])),
           tzFep Cos[angle6] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[2]]+primvecs [[3]])),
           tzFe Cos[angle5] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]])}}, ConstantArray[0, {3, 3}],
       ConstantArray[0, {3, 3}], ConstantArray[0, {3, 3}]},
      {{{tzFe Cos[angle5] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]]),
            tzFep Cos[angle6] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]]),
            tzFep Cos[angle6] ⅇ^(ⅈ kvec[kx,ky,kz].(-primvecs [[1]]-primvecs [[2]]+primvecs [[3]]))},
          {tzFep Cos[angle6] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]]),
            tzFe Cos[angle5] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]]),
            tzFep Cos[angle6] ⅇ^(ⅈ kvec[kx,ky,kz].(-primvecs [[1]]-primvecs [[2]]+primvecs [[3]]))},
          {tzFep Cos[angle6] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[2]]+primvecs [[3]])),
            tzFep Cos[angle6] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[2]]+primvecs [[3]])),
            tzFe Cos[angle5] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]])}}^†,
       ConstantArray[0, {3, 3}], ConstantArray[0, {3, 3}],
       ConstantArray[0, {3, 3}], ConstantArray[0, {3, 3}]},
      {ConstantArray[0, {3, 3}], ConstantArray[0, {3, 3}], tperpFe Cos[angle4]
         {{0, ⅇ^(-ⅈ kvec[kx,ky,kz].primvecs [[1]]), ⅇ^(-ⅈ kvec[kx,ky,kz].primvecs [[1]])},
          {ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[1]]), 0, 1}, {ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[1]]), 1, 0}},
       ConstantArray[0, {3, 3}], ConstantArray[0, {3, 3}]},
      {ConstantArray[0, {3, 3}], ConstantArray[0, {3, 3}], ConstantArray[0, {3, 3}],
       tperpFe Cos[angle4] {{0, ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[2]]), 1},
          {ⅇ^(-ⅈ kvec[kx,ky,kz].primvecs [[2]]), 0, ⅇ^(-ⅈ kvec[kx,ky,kz].primvecs [[2]])},
          {1, ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[2]]), 0}}, ConstantArray[0, {3, 3}]},
      {ConstantArray[0, {3, 3}], ConstantArray[0, {3, 3}], ConstantArray[0, {3, 3}],
       ConstantArray[0, {3, 3}], ConstantArray[0, {3, 3}]}}

hamiltonianFeFe [kx_, ky_, kz_, tzFe_, tzFep_, tperpFe_, CFS_] :=
  hamiltonianFeFeA [kx, ky, kz, tzFe, tzFep, tperpFe] +
   ArrayFlatten[Table[IdentityMatrix[3] CFS[[i, j]], {i, 5}, {j, 5}]]
```

Chromium-Iron sector defintion

```
In[90]:= hamiltonianCrFe[kx_, ky_, kz_, tCF_, tCFp_] :=
  ArrayFlatten@{{ConstantArray[0, {3, 3}], ConstantArray[0, {3, 3}],
      {{tCF Cos[ang3] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[3]])), tCF Cos[ang2]
            ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]]), tCF Cos[ang1] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]])},
        {tCF Cos[ang1] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[3]])), tCF Cos[ang3]
            ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]]), tCF Cos[ang2] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]])},
        {tCF Cos[ang2] ⅇ^(ⅈ (kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[3]]))), tCF Cos[ang1]
            ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]]), tCF Cos[ang3] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]])}},
      ConstantArray[0, {3, 3}], ConstantArray[0, {3, 3}]},
    {ConstantArray[0, {3, 3}], ConstantArray[0, {3, 3}],
      {{tCF Cos[ang3] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[1]]), tCF Cos[ang2], tCF Cos[ang1]},
        {tCF Cos[ang1] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[1]]), tCF Cos[ang3], tCF Cos[ang2]},
        {tCF Cos[ang2] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[1]]), tCF Cos[ang1], tCF Cos[ang3]}},
      ConstantArray[0, {3, 3}], ConstantArray[0, {3, 3}]},
    {ConstantArray[0, {3, 3}], ConstantArray[0, {3, 3}],
      ConstantArray[0, {3, 3}], {{tCF Cos[ang3] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]]),
          tCF Cos[ang2] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[2]]+primvecs [[3]])),
          tCF Cos[ang1] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]])},
        {tCF Cos[ang1] ⅇ^(ⅈ (kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[3]]))),
          tCF Cos[ang3] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[2]]+primvecs [[3]])),
          tCF Cos[ang2] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[3]]))},
        {tCF Cos[ang2] ⅇ^(ⅈ (kvec[kx,ky,kz].(-primvecs [[2]]+primvecs [[3]]))),
          tCF Cos[ang1] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]]),
          tCF Cos[ang3] ⅇ^(ⅈ kvec[kx,ky,kz].(-primvecs [[2]]+primvecs [[3]]))}},
      ConstantArray[0, {3, 3}]}, {ConstantArray[0, {3, 3}],
      ConstantArray[0, {3, 3}], ConstantArray[0, {3, 3}],
      {{tCF Cos[ang3], tCF Cos[ang2] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[2]]), tCF Cos[ang1]},
        {tCF Cos[ang1] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[1]]),
          tCF Cos[ang3] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[2]])),
          tCF Cos[ang2] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[1]])},
        {tCF Cos[ang2] ⅇ^(-ⅈ kvec[kx,ky,kz].primvecs [[2]]), tCF Cos[ang1],
          tCF Cos[ang3] ⅇ^(-ⅈ kvec[kx,ky,kz].primvecs [[2]])}},
      ConstantArray[0, {3, 3}]}, {{{tCFp Cos[ang4], tCFp Cos[ang4],
          tCFp Cos[ang5] ⅇ^(-ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[2]]))},
        {tCFp Cos[ang5] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[2]])),
          tCFp Cos[ang4] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[2]])), tCFp Cos[ang4]},
        {tCFp Cos[ang4] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[1]]), tCFp Cos[ang5]
            ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[1]]), tCFp Cos[ang4] ⅇ^(-ⅈ kvec[kx,ky,kz].primvecs [[2]])}},
      {{tCFp Cos[ang4] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]]),
          tCFp Cos[ang4] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]]),
          tCFp Cos[ang5] ⅇ^(ⅈ kvec[kx,ky,kz].(-primvecs [[1]]-primvecs [[2]]+primvecs [[3]]))},
        {tCFp Cos[ang5] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[2]]+primvecs [[3]])),
          tCFp Cos[ang4] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[2]]+primvecs [[3]])),
          tCFp Cos[ang4] ⅇ^(ⅈ kvec[kx,ky,kz].primvecs [[3]])},
        {tCFp Cos[ang4] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[3]])),
          tCFp Cos[ang5] ⅇ^(ⅈ kvec[kx,ky,kz].(primvecs [[1]]+primvecs [[3]])),
          tCFp Cos[ang4] ⅇ^(ⅈ kvec[kx,ky,kz].(-primvecs [[2]]+primvecs [[3]]))}},
      ConstantArray[0, {3, 3}], ConstantArray[0, {3, 3}], ConstantArray[0, {3, 3}]}}
```

Total Hamiltonian defintion

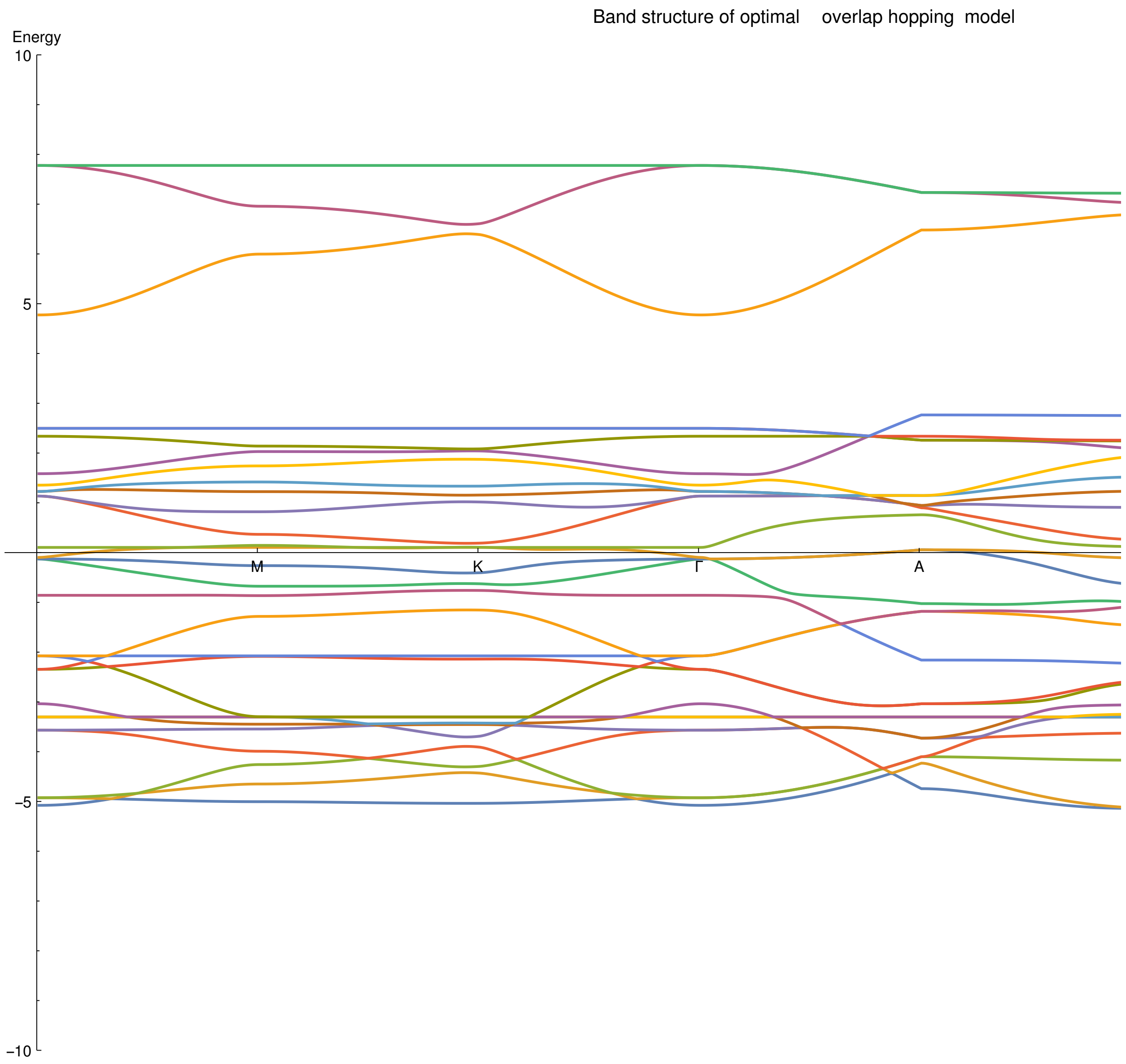
In[102]:=
```
hamiltonianFinal[kx_, ky_, kz_, tz_, tPerp_,
    tzp_, t2p_, tCF_, tCFp_, tzFe_, tzFep_, tperpFe_] :=
  ArrayFlatten@{{hamiltonianCrCr[kx, ky, kz, tz, tPerp, tzp, t2p, CFSCr],
       hamiltonianCrFe[kx, ky, kz, tCF, tCFp]},
     {hamiltonianCrFe[kx, ky, kz, tCF, tCFp]†,
       hamiltonianFeFe[kx, ky, kz, tzFe, tzFep, tperpFe, CFSFe]}}
```

Plot and export band structure

```
In[130]:= nElectrons = 33;
plotBandStructure[ham_ , title_, tz_,
    tPerp_, tzp_, t2p_, tCF_, tCFp_, tzFe_, tzFep_, tperpFe_] :=
  (
    hams  = ham [kPoints[[#, 1]], kPoints[[#, 2]], kPoints[[#, 3]], tz, tPerp, tzp ,
          t2p, tCF, tCFp, tzFe, tzFep, tperpFe] & /@Range@Length@kPoints;
    eigsList= Sort /@Chop@Eigenvalues/@hams  ;
    sorted = Sort@Flatten[eigsList];
    μ = sorted[[Round[(nElectrons / 2) Length@eigsList]]];
    data = Transpose[eigsList-μ];
    eigsys= Eigensystem /@hams  ;
    ListPlot[data, Joined→ True,
      PlotRange → {-10, 10}, PlotLabel → title, AxesLabel → {"", "Energy"},
      Ticks→ {{{0, "Γ"}, {nPoints, "M"}, {2 nPoints, "K"}, {3 nPoints, "Γ"},
            {4 nPoints, "A"}, {5 nPoints, "L"}, {6 nPoints, "H"},
            {7 nPoints, "A"}}, All} (*,PlotStyle→Black*)]
  )

plotMagneticBandStructure[ham_ , title_, tz_, tPerp_, tzp_, t2p_,
    tCF_, tCFp_, tzFe_, tzFep_, tperpFe_, JH_, SzCrvar_, SzFevar_] :=
  (
    hams  = ham [1, 1, 1, kPoints[[#, 1]], kPoints[[#, 2]],
          kPoints[[#, 3]], tz, tPerp, tzp , t2p, tCF, tCFp, tzFe, tzFep,
          tperpFe, 0, 0.05, JH, SzCrvar, SzFevar] & /@Range@Length@kPoints;
    eigsList= Sort /@Chop@Eigenvalues/@hams  ;
    sorted = Sort@Flatten[eigsList];
    μ = sorted[[Round[(nElectrons) Length@eigsList]]];
    data = Transpose[eigsList-μ];
    eigsys= Eigensystem /@hams  ;
    ListPlot[data, Joined→ True,
      PlotRange → {-8, 8}, PlotLabel → title, AxesLabel → {"", "Energy"},
      Ticks→ {{{0, "Γ"}, {nPoints, "M"}, {2 nPoints, "K"}, {3 nPoints, "Γ"},
            {4 nPoints, "A"}, {5 nPoints, "L"}, {6 nPoints, "H"}, {7 nPoints, "A"}}, All},
      (*PlotStyle→Black,*) ImageSize  → 800]
  )

exportBandStructure[ham_ , title_, tz_, tPerp_,
    tzp_, t2p_, tCF_, tCFp_, tzFe_, tzFep_, tperpFe_, name_ ] :=
  (
    bandStruct = plotBandStructure[ham  , title,
        tz, tPerp, tzp , t2p, tCF, tCFp, tzFe, tzFep, tperpFe];
    Export[name  , bandStruct]
  )

blackPlot = plotBandStructure[hamiltonianFinal,
      "Band Structure of effective Cr and Fe model ", 6, 1, 1, 1, 1, 1, 1, 1, 1];
```

```
In[134]:= blackPlot = plotBandStructure[hamiltonianFinal,
             "Band structure of optimal  overlap hopping model ", 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

Out[134]=


Band structure of optimal overlap hopping model
Energy
M
K
Γ
A