

Simple Penn Approximation

RafaCastle

Preambles

Energy loss function values and material properties

Here we import the energy loss function data (can't share those)

```
In[144]:= Needs["DifferentialEquations`InterpolatingFunctionAnatomy`"]  
          ⌈necesita  
          LaunchKernels[];  
          ⌈lanza kernels  
          ParallelEvaluate[Needs["DifferentialEquations`InterpolatingFunctionAnatomy`"]];  
          ⌈evalúa en paralelo ⌈necesita  
          kIDnums = ParallelEvaluate[$KernelID];  
                   ⌈evalúa en paralelo ⌈identificador de kernel  
          SetDirectory[NotebookDirectory[]]  
          ⌈establece direct... ⌈directorio de cuaderno
```

... **LaunchKernels:** Some subkernels are already running. Not launching default kernels again.

```
Out[148]:= C:\Users\chuch\Desktop\Guerda\Full Penn
```

```
In[149]:= NombreELF1 = "Al2O3 ELF.dat";  
          NombreELF2 = "CaF2 ELF.dat";  
          NombreELF3 = "LiF ELF.dat";  
          NombreELF4 = "H2O ELF.dat";  
          NombreC1 = "Al2O3 Cond.dat";  
          NombreC2 = "CaF2 Cond.dat";  
          NombreC3 = "LiF Cond.dat";  
          NombreC4 = "H2O Cond.dat";
```

```

In[157]:= Compuesto =
  ChoiceDialog["Escoge el compuesto", {Al2O3 → 1, CaF2 → 2, LiF → 3, Agua → 4, Otro → 5}];
  |diálogo de elección
Do[If[i == Compuesto, {temporal = Import[NombreELFi],
  |r... |si |importa
    Cond = Import[NombreCi], {bandgap, wmin, BVal, densidad, elfinicial} =
      |importa
      Table[Cond[[i]][[1]], {i, 1, Length[Cond]}], Break[]}], {i, 4}];
      |tabla |longitud |finaliza iteración
If[Compuesto == 5, {NombreELF5 = InputString[
  |si |cadena de caracteres de entrada
    "Ingrese el nombre del archivo que contiene la función de pérdida de energía",
    temporal = Import[NombreELF5], bandgap = Input["bandgap en eV"],
      |importa |entra
    wmin = Input["ω min en eV"], BVal = Input["Ancho de banda de valencia en eV"],
      |entra |entra
    densidad = Input["Densidad en g/cm^3"], elfinicial = 1}];
      |entra
Do[temporal[[i, 1]] = temporal[[i, 1]] * QuantityMagnitude[UnitConvert["eV", "Hartrees"]],
  |repite |magnitud de cantidad |convierte unidad
  {i, 1, Length[temporal]}];
  |longitud
ELFData = temporal[[1 ;; All]];
  |todo

c = 137;
bandgap = bandgap * QuantityMagnitude[UnitConvert["eV", "Hartrees"]];
  |magnitud de cantidad |convierte unidad
BVal = BVal * QuantityMagnitude[UnitConvert["eV", "Hartrees"]];
  |magnitud de cantidad |convierte unidad
wmin = wmin * QuantityMagnitude[UnitConvert["eV", "Hartrees"]];
  |magnitud de cantidad |convierte unidad
inicial = elfinicial;

```

Above I defined some physical constants and converted them to Hartree system

Interpolations

Plot of the approximate energy loss function

```

In[ ]:= ELF = Interpolation[Join[{0, 0}], ELFData], InterpolationOrder → 1];
  |interpolación |junta |orden de interpolación

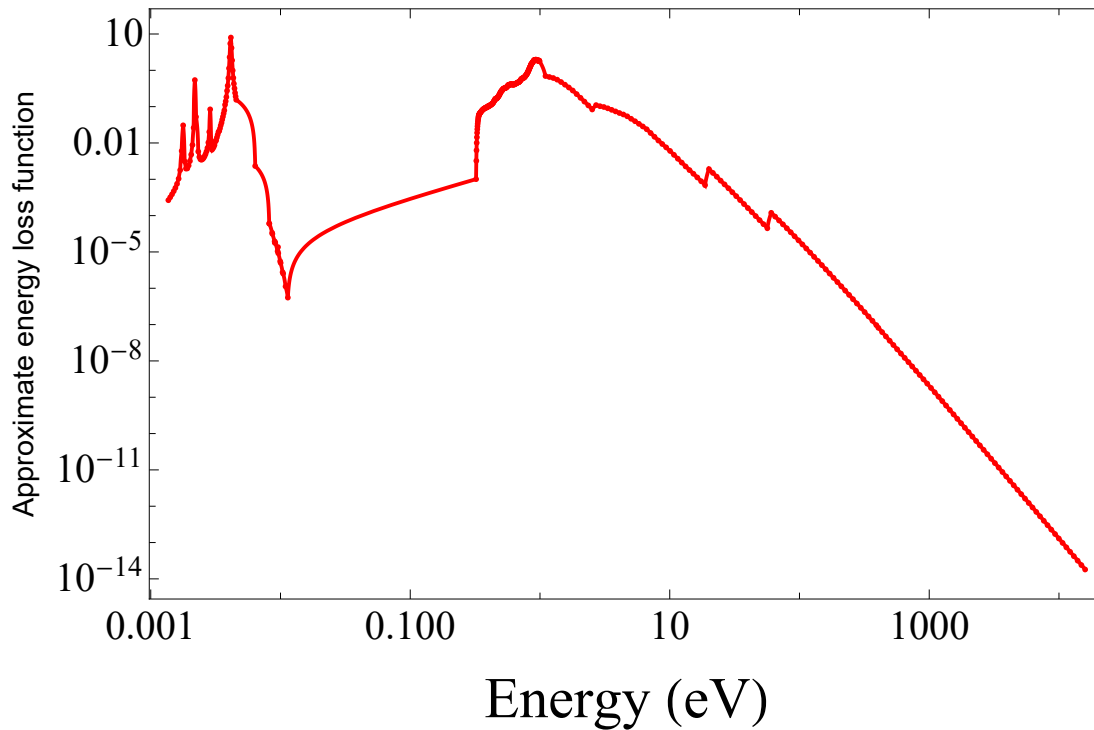
```

```

In[191]:= TL = 30;
TLe = 21;
TLet = 20;
TPL = 10;
LogLogPlot[ELF[w], {w, First[ELFData][[1]], Last[ELFData][[1]]},
  representación log log primero último
  PlotRange → All, Frame → True, PlotStyle → {Thick, Red},
  rango de rep... todo marco verd... estilo de repre... grueso rojo
  Epilog → {Red, PointSize[0.006], Point[Log /@ ELFData]}, PlotRange → All,
  epílogo rojo tamaño de punto punto logaritmo rango de rep... todo
  ImageSize → Large, FrameTicksStyle → Directive[Black, TLe, FontFamily → "Times"],
  tamaño de i... grande estilo de marcas del m... directiva negro familia de tipo de... multiplicació
  FrameLabel → {Style["Energy (eV)", FontFamily → "Times", TL],
  etiqueta de marco estilo familia de tipo de... multiplicación
  Style["Approximate energy loss function", 0.5 TL]}]
estilo
g[w_] := (2 / (Pi * w)) * ELF[w]
número pi

```

Out[195]=



Equations

```
In[ ]:= kf[wp_] := ((3 * Pi / 4) ^ (1 / 3)) * (wp) ^ (2 / 3);
           |_número pi
Ef[wp_] := (kf[wp] ^ 2) / 2
Vf[wp_] := Sqrt[2 * Ef[wp]]
           |_raíz cuadrada
wq[q_, wp_] := Sqrt[wp^2 + ((1 / 3) * (q^2) * ((Vf[wp]) ^ 2)) + (q^4) / 4]
           |_raíz cuadrada
```

Integration limits

Obtengamos las componentes de la energía.

```
In[ ]:= coords1 = First[InterpolatingFunctionCoordinates[ELF]];
           |_primero
final = Length[coords1];
           |_longitud
```

Se define ahora la ecuación 2 del artículo de Guerda y Miguel, definiendo primero los límites de integración y a T'.

```
In[ ]:= T[Ei_] := Ei;
Tp[Ei_] := T[Ei] - bandgap;
qm[T_, w_] := Sqrt[Tp[T] (2 + Tp[T] / (c^2))] - Sqrt[(Tp[T] - w) (2 + (Tp[T] - w) / (c^2))]
           |_raíz cuadrada |_raíz cuadrada
qp[T_, w_] := Sqrt[Tp[T] (2 + Tp[T] / (c^2))] + Sqrt[(Tp[T] - w) (2 + (Tp[T] - w) / (c^2))]
           |_raíz cuadrada |_raíz cuadrada
```

Dada Tp, T debe ser mayor a bandgap + BVal (26 p el Al2O3)

```
In[ ]:= Fac[T_] := (((1 + Tp[T] / (c^2)) ^ 2) / (1 + Tp[T] / (2 c^2))) (1 / (Pi * Tp[T]))
           |_número pi
qm1[w_, wp_] := -kf[wp] + Sqrt[(kf[wp]) ^ 2 + 2 * w]
           |_raíz cuadrada
qp1[w_, wp_] := kf[wp] + Sqrt[(kf[wp]) ^ 2 + 2 * w]
           |_raíz cuadrada
```

Numeric method

Variables ω and q .

```
In[ ]:= inicial = elfinicial
final = Length[coords1]
           |_longitud
```

Out[]:= 310

Out[]:= 454

```

In[ ]:= sq = 20;
sw = 20;
s[v_] := 1/v
kw = 0;
kq = 0;
cs = 0;
ccs = 1;
Do[{pw = Tp[coords1[[kT]]] - BVal - wmin,
  repite
  While[wmin + s[sw] * kw * pw ≤ Tp[coords1[[kT]]] - BVal,
    mientras
    {vwkT,kw = wmin + s[sw] * kw * pw, pq = qp[coords1[[kT]]], vwkT,kw - qm[coords1[[kT]]], vwkT,kw},
    While[qm[coords1[[kT]]], vwkT,kw + s[sq] * kq * pq ≤ qp[coords1[[kT]]], vwkT,kw},
    mientras
    vqkT,kw,kq = qm[coords1[[kT]]], vwkT,kw + s[sq] * kq * pq;
    kq++, qfinkT,kw = kq - 1, kq = 0};
    kw++, wfinkT = kw - 1, kw = 0, If[cs == 90,
      si
      {sq = sq + 5, sw = sw + 10, ccs = ccs + 1, cs = 80 + ccs}, cs = cs + 1]}, {kT, inicial, final}]

```

Out[]:= \$Aborted

```

In[ ]:= Do[{EEWkT = wfinkT, EEQkT,kw = qfinkT,kw}, {kT, inicial, final}, {kw, 0, wfinkT}]
  repite

```

```

In[ ]:= sq = 5;
sw = 100; (*cambiar a 100*)
s[v_] := 1/v
kw = 1;
kq = 0;
cs = 0;
ccs = 1;
Do[{pw = vwkT,1 - vwkT,0,
  repite
  While[vwkT,0 + s[sw] * kw * pw < vwkT,1, {vwkT,EEWkT+kw = vwkT,0 + s[sw] * kw * pw,
    mientras
    pq = qp[coords1[[kT]]], vwkT,EEWkT+kw - qm[coords1[[kT]]], vwkT,EEWkT+kw}},
    While[qm[coords1[[kT]]], vwkT,EEWkT+kw + s[sq] * kq * pq ≤ qp[coords1[[kT]]], vwkT,EEWkT+kw}},
    mientras
    vqkT,EEWkT+kw,kq = qm[coords1[[kT]]], vwkT,EEWkT+kw + s[sq] * kq * pq;
    kq++, qfinkT,EEWkT+kw = kq - 1, kq = 0};
    kw++, wfinkT = EEWkT + kw - 1, kw = 1}, {kT, inicial, final}]

```

WO

```

Do[{If[q == 0, wr = 0.9 * vwi,w], w0i,w,q = FindRoot[wq[vqi,w,q, wp] - vwi,w == 0, {wp, wr}][[1, 2]],
  rep··[si
    wr = 0.9 * w0i,w,q, If[q == qfini,w && w == wfini, Print[{i}]],
    [si [pertenece a [números··[continúa iteración
      If[Element[w0i,w,q, Reals], Continue, w0i,w,q = 0]],
      {i, inicial, final}, {w, 0, wfini}, {q, 0, qfini,w}]
    Do[{vPLkT,kw,kq = Re[(w0kT,kw,kq / vwkT,kw)] * ELF[Re[w0kT,kw,kq] ]],
      [parte real [parte real
        (*If[kw==wfinkT && kq==qfinkT,kw, Print[kT]], *)
        [si [escribe
          If[Element[vPLkT,kw,kq, Reals], Continue, Print[{kT, kw, kq}]]],
          [pertenece a [números··[continúa it·· [escribe
            {kT, inicial, final}, {kw, 0, wfinkT}, {kq, 0, qfinkT,kw}]

```


ELF

```

In[112]:= IMEPLS = Interpolation[Flatten[Table[{vwkT,kw, vqkT,kw,kq}, vPLkT,kw,kq},
  [interpolación [aplana [tabla
    {kT, inicial, final}, {kw, 0, wfinkT}, {kq, 0, qfinkT,kw}], 2], InterpolationOrder -> 1]
  [orden de interpolación

IMEPL[w_, q_] := IMEPLS[
  w,
  q]

```

Out[112]= InterpolatingFunction[
 Domain: {{0.317, 1.59 × 10⁴}, {0.00275, 426.}}
 Output: scalar
 Los datos no están en el cuaderno; almacénelos ahora mismo »

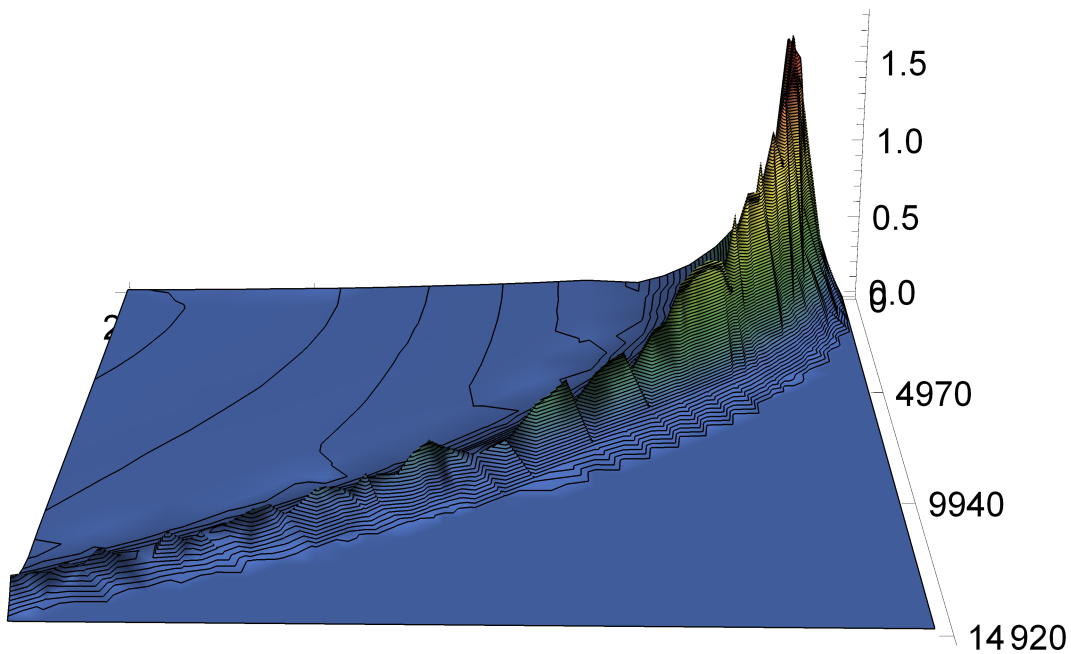
The IMEPLS(w,q) interpolation corresponds to the energy loss function.

```

In[139]:= wmax = 8 (*InterpolatingFunctionDomain[IMEPLS][[1,2]]*);
qmax = 4 (*InterpolatingFunctionDomain[IMEPLS][[2,2]]*);
wc = QuantityMagnitude[UnitConvert["Hartrees", "eV"]];
      [magnitud de cantidad] [convierte unidad]
qc = 1.99285191410 * 10^(-24) * QuantityMagnitude[UnitConvert["Kg", "eV/c^2"]] /
      [magnitud de cantidad] [convierte unidad]
      QuantityMagnitude[UnitConvert["c", "m/s"]];
      [magnitud de cantidad] [convierte unidad]
Plot3D[IMEPL[w, q], {w, InterpolatingFunctionDomain[IMEPLS][[1, 1]], wmax},
[representación gráfica 3D]
  {q, 0.05, qmax}, PlotRange -> All, LabelStyle -> Directive[18], PlotRange -> All,
      [rango de rep... [todo [estilo de etiqueta [directiva [rango de rep... [todo
Mesh -> 100, MeshFunctions -> {#3 &}, MeshStyle -> {Black}, MeshShading -> {Automatic},
[malla [funciones de divisiones de malla [estilo de malla [negro [sombreado de ma... [automático
Boxed -> False, AxesEdge -> {{-1, -1}, {-1, -1}, {-1, -1}}, TicksStyle -> Directive[Black],
[rodead... [falso [borde de ejes [estilo de marcas [directiva [negro
Ticks -> {Table[{i * (wmax / 4), Round[i * wc * (wmax / 4), 5]}, {i, 0, 4}],
[marcas [tabla [entero más próximo
      Table[{i * qmax / 3, Round[i * (qmax / 3) * qc, 10]}, {i, 0, 3}], Automatic},
      [tabla [entero más próximo [automático
ColorFunction -> "DarkRainbow", ImageSize -> Large]
[función de color [tamaño de im... [grande

```

Out[143]=



Stopping power

```

In[ ]:= PF[kT_?NumericQ] := Fac[coords1[[kT]]] *
  expresión numérica?
  NIntegrate[(1/qq) * ww * (IMEPL[ww, qq]), {ww, wmin, Tp[coords1[[kT]]] - BVal},
  integra numéricamente
  {qq, qm[coords1[[kT]], ww], qp[coords1[[kT]], ww]},
  AccuracyGoal → 20, MinRecursion → 4, MaxRecursion → 150]
  objetivo de exactitud recursión mínima máxima recursión

In[ ]:= Do[{PdfkT = PF[kT], Print[kT]}, {kT, inicial, final, 2}]
  repite escribe

In[ ]:= PodFr = Table[
  tabla
  {(coords1[[kT]] - bandgap - BVal) * QuantityMagnitude[UnitConvert["Hartrees", "eV"]],
  magnitud de cantidad convierte unidad
  ((PdfkT * QuantityMagnitude[UnitConvert["Hartrees", "eV"]]) / QuantityMagnitude[
  magnitud de cantidad convierte unidad magnitud de cantidad
  UnitConvert["BohrRadius", "nanometers"]]}, {kT, inicial, final, 2}];
  convierte unidad

ListLogLogPlot[PodFr, PlotStyle → {Blue, Thick}, Frame → True,
  representación log log de lista estilo de repre... azul grueso marco verdadero
  FrameLabel → {"Energía (eV)", "Poder de frenado lineal de colisión (eV/nm)"},
  etiqueta de marco
  PlotRange → All, PlotLegends → Placed[{"Este trabajo"}, {Right, Top}]
  rango de rep... todo leyendas de rep... colocado derecha arriba

```

