```
In[•]:= x=.; Remove["Global`*"];
        Off[General::spell1];

In[•]:= conjugateRule={Complex[re_,im_]:>Complex[re,-im]};
        conjugate[z_]:=z/.conjugateRule;
        real[z_]:=(1/2)*(z+conjugate[z])//Simplify;
        imag[z_]:=(-I/2)*(z-conjugate[z])//Simplify;
        abs[z_]:=Sqrt[(z*(z/.conjugateRule))]//Simplify;
```

simulation

Constants

```
In[•]:= c = 299 792 458 ;
        hb = 1.0545715964207855 * 10^-34;
        q = 1.602 * 10^-19;
        ϵ0 = 8.85 * 10^-12;
        m = 9.1 * 10^-31;
        a0 = 4.0494 * 10^-10;
        den = 4 / a0^3;
        G0 = Sqrt[(2^2) + (0) + (0)] * 2 * Pi / a0;
        normstructurefactor = 1;
        atomscatfactor = 5;
        re = 2.82 * 10^-13 * 10^-2;
        ϵ0 = 8.85 * 10^-12;
        m = 9.1 * 10^-31;

In[•]:= heV = 4.135667662 * 10^-15;

In[•]:= atomscatfactor * normstructurefactor * 4
Out[•]=
        20

In[•]:=
```

Scattering Factors (Aluminum). Data is hidden in this cell

SPP dispersion vs dielectric light line

```
γ = 0.12 * wp;
θB400 = Abs[ArcSin[2 * G0 / (2 * kp0)]];
θs0 = (θB400 * 180 / Pi - 0.04) * Pi / 180;

Ep0 = 9978;
wp0 = Ep0 * q / hb;
kp0 = wp0 / c * n[wp0];
G = G0;
```

```
dBShift = 0.02;
```

```
ip = 9978;
```

$$kx[w\_] := \frac{w}{c}\left(\sqrt{\frac{epsM[w]}{epsM[w]+1}}\right)$$

```
γ = 0.12 * wp;
```

```
kp0 = n[ip * (q / hb)] * ip * (q / hb) / c;
```

```
ks[i_] := n[ip * (q / hb)] (ip - i) * (q / hb) / c;
```

```
kix[i_] := Re[kx[i * q / hb]]
```

```
kpx[i_, dB_] := ks[i] * Cos[θs0 - (dB + dBShift) * Pi / 180] - kix[i]
```

$$kpxLightLine[i\_, dB\_] := ks[i] * Cos[\theta s0 - (dB + dBShift) * Pi / 180] - \frac{i * (q / hb)}{c}$$

```
θp[i_, dB_] := ArcCos[kpx[i, dB] / kp0]
```

```
θpLightLine[i_, dB_] := ArcCos[kpxLightLine[i, dB] / kp0]
```

Applying Newton's method to solve for the angular SPP dispersion

```
initialGuess = 0.15;
(* You may need to adjust the initial guess based on the expected solution range *)

solution = FindRoot[
  dB == -dBShift + (ArcCos[kpx[10, dB] / kp0] - θB400) * 180 / Pi, {dB, initialGuess}]
```

Out[•]=

```
{dB → 0.0672412}
```

```
bla = dB /. (FindRoot[
    dB == -dBShift + (ArcCos[kpx[10, dB] / kp0] - θB400) * 180 / Pi, {dB, initialGuess}])
```

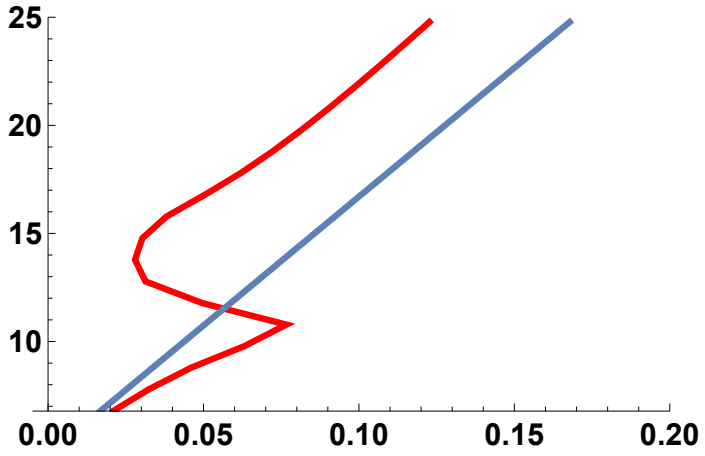
Out[•]=

```
0.0672412
```

```
initialGuess = 0.05;
Ei1 = 6.78;
Ei2 = 25;

Show[ListPlot[Table[
    {dB /. (FindRoot[dB == -dBShift + (θp[i, dB] - θB400) * 180 / Pi, {dB, initialGuess}]),
     i}, {i, Ei1, Ei2}], Joined → True,
  PlotRange → {{-0.005, 0.20}, {Ei1, Ei2}}, PlotStyle → {Red, Thickness[0.01]},
  PlotLabel → None, LabelStyle → {16, GrayLevel[0], Bold}],
 ListPlot[Table[{dB /. (FindRoot[dB == -dBShift + (θpLightLine[i, dB] - θB400) * 180 / Pi,
       {dB, initialGuess}]), i}, {i, Ei1, Ei2}],
  Joined → True, PlotRange → {{-0.005, 0.20}, {Ei1, Ei2}},
  PlotStyle → {(*Black,Dashed,*)Thickness[0.01]},
  PlotLabel → None, LabelStyle → {16, GrayLevel[0], Bold}]]
```

Out[•]=



QED simulation with γ=0.12*wp

In[•]:= $\mu 0 = \frac{1}{c^2\, \epsilon 0}$;

$\eta 0$ = Sqrt[μ0 / ϵ0]

Out[•]=

376.909

In[•]:= ip = 9978;

In[•]:= kp0 = n[ip * (q / hb)] * ip * (q / hb) / c;

In[•]:= ks[i_] := n[ip * (q / hb)] (ip - i) * (q / hb) / c;

```
In[•]:= γ = 0.12 * wp;
        θB400 = Abs[ArcSin[2 * G0 / (2 * kp0)]];
        θs0 = (θB400 * 180 / Pi - 0.04) * Pi / 180;
        Lz = 3 * 10^-8;
        xs = 0;
        dBShift = 0(*0.048*);

        j0 = 25
        Ep0 = 9978;
        wp0 = Ep0 * q / hb;
        wi0 = j0 * q / hb;
        ws0 = wp0 - (j0 * q / hb);
        kp0 = wp0 / c * n[wp0];
        ks0 = ws0 / c * n[ws0];
        ki0 = wi0 * n[wi0] / c;
        G = G0;

        θB = Abs[ArcSin[2 * G0 / (2 * kp0)]];
Out[•]=
        25

In[•]:= Re[k2M[25 * q / hb]]
Out[•]=
        9.83998 × 10^7

In[•]:= Abs[k2M[25 * q / hb]]
Out[•]=
        9.84304 × 10^7

In[•]:= βs[j_, dB_, xs_] := η0 / (2 * hb * (wp0 - (j * q / hb))) 1 / (Cos[θs0 + xs - (dB + dBShift) * Pi / 180])^2;

In[•]:= βs[22]
Out[•]=
        βs[22]

In[•]:= idlerNorm[j_] := hb * (j * q / hb)^2 / (Pi * ϵ0 * c^2);

In[•]:= idlerNorm[20]
Out[•]=
        3.8956 × 10^-8

In[•]:= pumpPreFactor = 2 * hb * η0 * wp0 / Sin[θB400];

In[•]:= ρG = q * (den * 3 * atomscatfactor * normstructurefactor);

In[•]:= σs[j_] := q^2 * ρG / (m^2 * wp0 * (j * q / hb)^2);

In[•]:= σs[22]
Out[•]=
        2.64994 × 10^-19
```

In[•]:=
```
ws[j_] := wp0 - (j * q / hb);
```

In[•]:=
```
ηs = 1/(ϵ0 * c); ηp = 1/(ϵ0 * c)
```

Out[•]=
```
376.909
```

In[•]:=
```
χG[j_] := (q * ρG)/(ϵ0 * (j * q / hb)^2);
```

In[•]:=
```
σNLSquared[j_] := ((0.01)^2 * q^2 * ϵ0^2 * (2 * G0)^2 (Cos[2 * θB])^2)/(4 * m^2 * (ws[j])^2);
```

In[•]:=
```
(βs[10, 0, 0]) * pumpPreFactor * (σs[10])^2 * (idlerNorm[10]) * (2 * G0)^2 * 10^13 * (ks[10])^2
```

Out[•]=
$$5.84605 \times 10^{17}$$

In[•]:=
```
flux0 = 10^13;(* at 10 KeV per 0.1% BW *)
```

In[•]:=
```
BW0 = 0.1 * 0.01; (*per 0.1% BW. This is equivalent to 10eV/10KeV*)
```

In[•]:=
```
BWmonochromator = N[2 / 10000];(* 2eV/10KeV *)
```

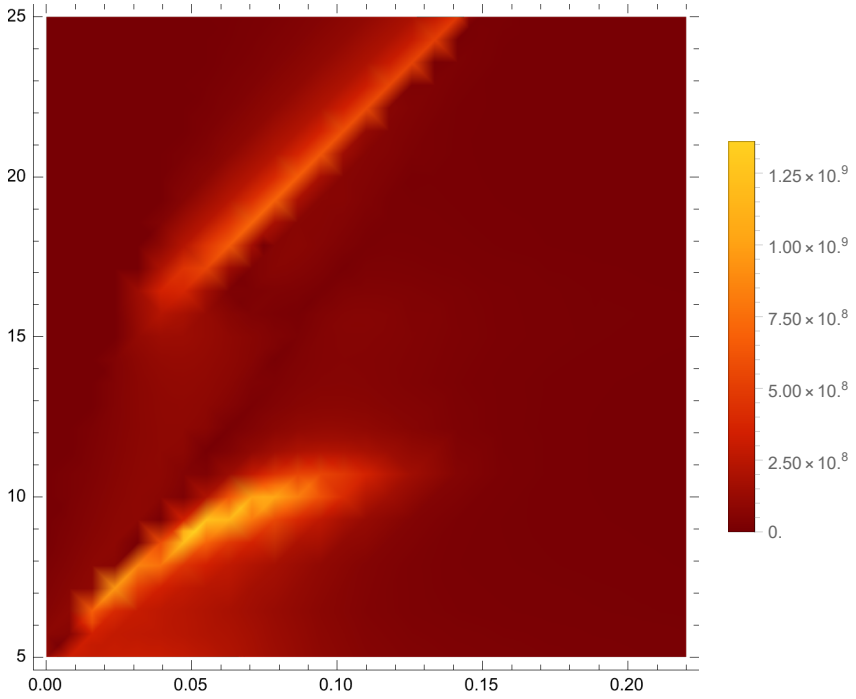
$$\text{PositiveSolutions} = \left(\frac{1 - \text{Erf}\left[\frac{\text{qiiXZ}+\frac{2*q/hb}{c}}{0.0005\,\frac{25*q/hb}{c}}\right]}{2}\right);$$

$$\text{ScatteringDumping} = \left(\frac{1 - \text{Erf}\left[\frac{\text{qiiXZ}+\frac{2*q/hb}{c}}{0.0005\,\frac{25*q/hb}{c}}\right]}{2}\right)\frac{1 + \text{Erf}\left[\frac{\text{qiiXZ}+\frac{21*q/hb}{c}}{0.3\,\frac{25*q/hb}{c}}\right]}{2};$$

```
func[dB_, i_, ii_, xs_, ys_, θg_] :=
 ((q / hb) (βs[i + ii, dB, xs]) * pumpPreFactor * (σs[i + ii])^2 *
    (idlerNorm[i + ii]) * (2 * G0)^2 * (flux0 * (BWmonochromator / BW0)) * (ks[i + ii])^2 *
    Cos[θs0 + xs - (dB + dBShift) * Pi / 180] * ((Abs[qii] / Abs[k2M[(i + ii) * q / hb]])^2) *
    (Abs[PositiveSolutions * (Re[1 / k2Mz[(i + ii) * q / hb, qii]] Abs[
          (1 - e^(I (dkzPS + Re[k2Mz[(i+ii)*q/hb,qii]]) Lz - Im[k2Mz[(i+ii)*q/hb,qii]] Lz)) /
           (I (dkzPS + Re[k2Mz[(i + ii) * q / hb, qii]]) - Im[k2Mz[(i + ii) * q / hb, qii]])]^2 .
       Re[1 / k2Mz[(i + ii) * q / hb, qii]] Abs[(1 -
          e^(-I (dkzPS - Re[k2Mz[(i+ii)*q/hb,qii]]) Lz - Im[k2Mz[(i+ii)*q/hb,qii]] Lz)) /
         (-I (dkzPS - Re[k2Mz[(i + ii) * q / hb, qii]]) -
           Im[k2Mz[(i + ii) * q / hb, qii]])]^2) +
      ScatteringDumping * Re[(rpM[(i + ii) * q / hb, qii] / k2Mz[(i + ii) * q / hb, qii])
        (1 - e^(I (dkzPS + Re[k2Mz[(i+ii)*q/hb,qii]]) Lz - Im[k2Mz[(i+ii)*q/hb,qii]] Lz)) /
         (I (dkzPS + Re[k2Mz[(i + ii) * q / hb, qii]]) - Im[k2Mz[(i + ii) * q / hb, qii]])
        (1 - e^(-I (dkzPS - Re[k2Mz[(i+ii)*q/hb,qii]]) Lz - Im[k2Mz[(i+ii)*q/hb,qii]] Lz)) /
         (-I (dkzPS - Re[k2Mz[(i + ii) * q / hb, qii]]) - Im[k2Mz[(i + ii) * q / hb, qii]])]])) /.
 {(*j→i+ii,*) qiiXZ → (kp0 * Cos[-(θB400 * 180 / Pi + (dB + dBShift)) * Pi / 180] -
     2 * G0 * Sin[θg * Pi / 180] - ks[i + ii] * Cos[θs0 + xs - (dB + dBShift) * Pi / 180]),
  qii → (*(kp0*Cos[-(θB400*180/Pi+(dB+dBShift))*Pi/180]-
     ks[i+ii]*Cos[θs0+xs-(dB+dBShift)*Pi/180])*) Sqrt[
    (kp0 * Cos[-(θB400 * 180 / Pi + (dB + dBShift)) * Pi / 180] - 2 * G0 * Sin[θg * Pi / 180])^
      2 - 2 * (kp0 * Cos[-(θB400 * 180 / Pi + (dB + dBShift)) * Pi / 180] -
        2 * G0 * Sin[θg * Pi / 180]) * (ks[i + ii] * Cos[θs0 + xs - (dB + dBShift) * Pi / 180]) *
      Cos[ys] + (ks[i + ii] * Cos[θs0 + xs - (dB + dBShift) * Pi / 180])^2],
  dkzPS → kp0 * Sin[-(θB400 * 180 / Pi + (dB + dBShift)) * Pi / 180] +
    2 * G0 * Cos[θg * Pi / 180] - ks[i + ii] * Sin[θs0 + xs - (dB + dBShift) * Pi / 180]}
```

In[•]:=
```
DensityPlot[func[dB, i, 0, 0, 0, 0.021], {dB, 0, 0.22}, {i, 5, 25}, PlotRange → All,
 PlotLegends → Automatic, PerformanceGoal → "Quality", ColorFunction → "SolarColors"]
```

Out[•]=



In[•]:=
```
Clear[dataMap1];
dataMap1 = ParallelTable[
  {dB, i, NIntegrate[Abs[func[dB, i, ii, xs, ys, 0.021]],
    {ii, -0.75, 0.75}, {xs, -6 * 10^-4, 6 * 10^-4}, {ys, -6 * 10^-4, 6 * 10^-4},
    Method → {"QuasiMonteCarlo", "MaxPoints" → 1 * 10^3}]},
  {dB, -0.005, 0.22, 0.01}, {i, 6.78, 25, 0.2}]
```

(kernel 2)
```
NIntegrate::maxp :
 The integral failed to converge after 1000 integrand evaluations. NIntegrate obtained
   74.76441861218608` and 1.0655521516939603` for the integral and error estimates.
```

(kernel 2)
```
NIntegrate::maxp :
 The integral failed to converge after 1000 integrand evaluations. NIntegrate obtained
   75.3600441067054` and 1.1219418814819908` for the integral and error estimates.
```

(kernel 2)
```
NIntegrate::maxp :
 The integral failed to converge after 1000 integrand evaluations. NIntegrate obtained
   76.83256973402922` and 1.1978876463967127` for the integral and error estimates.
```

(kernel 2)
```
General::stop :
 Further output of NIntegrate::maxp will be suppressed during this calculation.
```

(kernel 1)
```
NIntegrate::maxp :
 The integral failed to converge after 1000 integrand evaluations. NIntegrate obtained
   25.215185228812427` and 0.2674192845978723` for the integral and error estimates.
```

(kernel 1)

```
NIntegrate::maxp :
 The integral failed to converge after 1000 integrand evaluations. NIntegrate obtained
   25.076005848234516` and 0.28839288344157227` for the integral and error estimates.
```

(kernel 1)

```
NIntegrate::maxp :
 The integral failed to converge after 1000 integrand evaluations. NIntegrate obtained
   25.235670092502495` and 0.3186060916891115` for the integral and error estimates.
```

(kernel 1)

```
General::stop :
 Further output of NIntegrate::maxp will be suppressed during this calculation.
```

(kernel 4)

```
NIntegrate::maxp :
 The integral failed to converge after 633 integrand evaluations. NIntegrate obtained
   10.972166547964273` and 0.11031914108620752` for the integral and error estimates.
```

(kernel 4)

```
NIntegrate::maxp :
 The integral failed to converge after 742 integrand evaluations. NIntegrate obtained
   10.963291451073333` and 0.11250228843827098` for the integral and error estimates.
```

(kernel 4)

```
NIntegrate::maxp :
 The integral failed to converge after 907 integrand evaluations. NIntegrate obtained
   11.030758743061007` and 0.11408571759215483` for the integral and error estimates.
```

(kernel 4)

```
General::stop :
 Further output of NIntegrate::maxp will be suppressed during this calculation.
```

(kernel 3)

```
NIntegrate::maxp :
 The integral failed to converge after 1000 integrand evaluations. NIntegrate obtained
   9.10588713991264` and 0.0974390178393471` for the integral and error estimates.
```

(kernel 3)

```
NIntegrate::maxp :
 The integral failed to converge after 1000 integrand evaluations. NIntegrate obtained
   8.15252327215074` and 0.09038907408852463` for the integral and error estimates.
```

(kernel 3)

```
NIntegrate::maxp :
 The integral failed to converge after 1000 integrand evaluations. NIntegrate obtained
   7.251050042025095` and 0.08099479818150714` for the integral and error estimates.
```

(kernel 3)

```
General::stop :
 Further output of NIntegrate::maxp will be suppressed during this calculation.
```

Out[•]=

{{{-0.005, 6.78, 97.665}, {-0.005, 6.98, 75.1497}, {-0.005, 7.18, 60.6931},
  {-0.005, 7.38, 53.6788}, {-0.005, 7.58, 48.6526}, {-0.005, 7.78, 44.1162}, {-0.005, 7.98, 39.8721},
  {-0.005, 8.18, 35.9357}, {-0.005, 8.38, 32.4031}, {-0.005, 8.58, 29.1951}, {-0.005, 8.78, 26.0547},
  ··· 70 ···, {-0.005, 22.98, 0.}, {-0.005, 23.18, 0.}, {-0.005, 23.38, 0.}, {-0.005, 23.58, 0.},
  {-0.005, 23.78, 0.}, {-0.005, 23.98, 0.}, {-0.005, 24.18, 0.}, {-0.005, 24.38, 0.},
  {-0.005, 24.58, 0.}, {-0.005, 24.78, 0.}, {-0.005, 24.98, 0.}}, ··· 21 ···, {··· 1 ···}}

Full expression not available (original memory size: 255.6 kB)

```
In[•]:= Clear[dataMap2];
        dataMap2 = Partition[Flatten[dataMap1], 3]
```

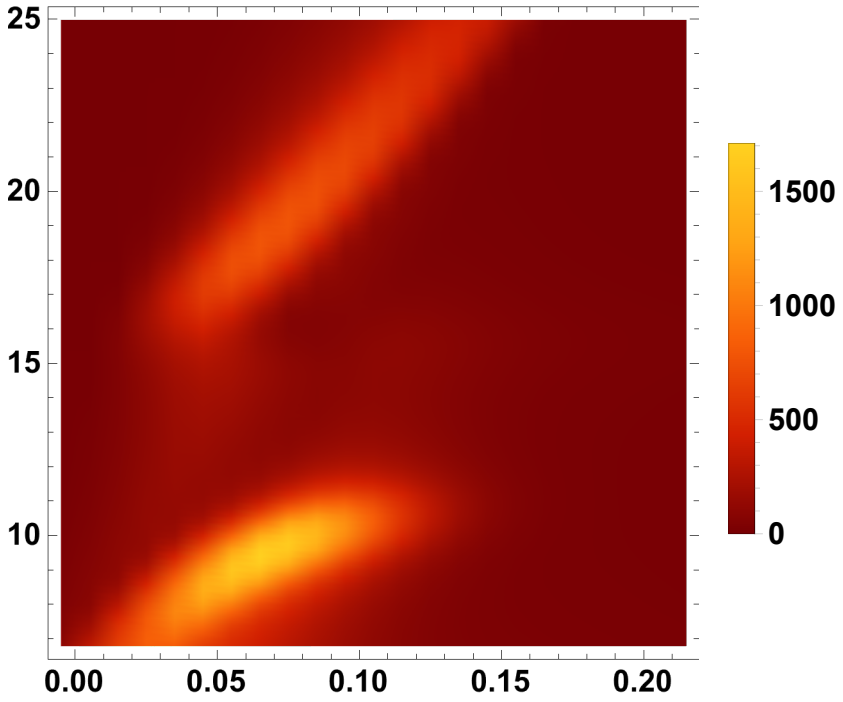
Out[•]=

```
{{-0.005, 6.78, 97.665}, {-0.005, 6.98, 75.1497}, {-0.005, 7.18, 60.6931},
 {-0.005, 7.38, 53.6788}, {-0.005, 7.58, 48.6526}, {-0.005, 7.78, 44.1162}, {-0.005, 7.98, 39.8721},
 {-0.005, 8.18, 35.9357}, {-0.005, 8.38, 32.4031}, ··· 2098 ···, {0.215, 23.38, 0.891395},
 {0.215, 23.58, 0.874436}, {0.215, 23.78, 0.859428}, {0.215, 23.98, 0.846296}, {0.215, 24.18, 0.834982},
 {0.215, 24.38, 0.825443}, {0.215, 24.58, 0.817653}, {0.215, 24.78, 0.8116}, {0.215, 24.98, 0.807292}}
```

Full expression not available (original memory size: 254.4 kB)

```
In[•]:= ListDensityPlot[dataMap2, PlotRange → All, PlotLegends → Automatic,
         ColorFunction → "SolarColors", PlotLabel → None, LabelStyle → {16, GrayLevel[0], Bold}]
```

Out[•]=




```
In[•]:= N[2700 / 200]
```

Out[•]=

```
13.5
```

2D dispersion plot on top of QED plot

```
In[•]:= γ = 0.12 * wp;
```

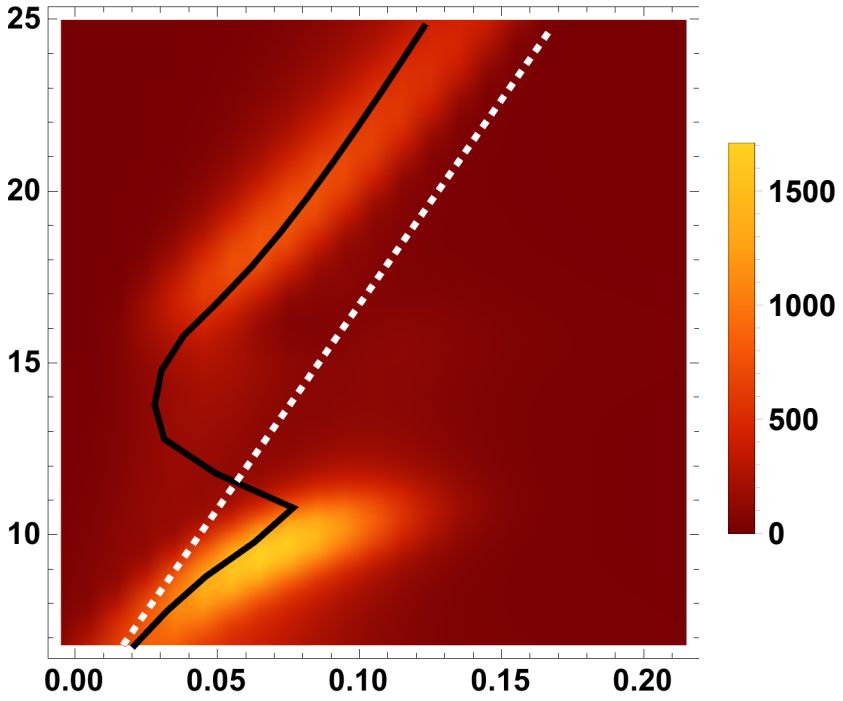
```
In[•]:= dBShift = 0.02;
```

In[•]:=
```
Show[ListDensityPlot[dataMap2, PlotRange → All,
  PlotLegends → Automatic, ColorFunction → "SolarColors", PlotLabel → None,
  LabelStyle → {16, GrayLevel[0], Bold}], ListPlot[Table[
   {dB /. (FindRoot[dB == -dBShift + (θp[i, dB] - θB400) * 180 / Pi, {dB, initialGuess}]),
    i}, {i, Ei1, Ei2}], Joined → True, PlotRange → {{-0.005, dBmax}, {Ei1, 25}},
  PlotStyle → {Black, Thickness[0.01]}, PlotLabel → None,
  LabelStyle → {16, GrayLevel[0], Bold}],
 ListPlot[Table[{dB /. (FindRoot[dB == -dBShift + (θpLightLine[i, dB] - θB400) * 180 / Pi,
       {dB, initialGuess}]), i}, {i, Ei1, Ei2}],
  Joined → True, PlotRange → {{-0.005, dBmax}, {Ei1, Ei2}},
  PlotStyle → {White, Dashed, Thickness[0.01]},
  PlotLabel → None, LabelStyle → {16, GrayLevel[0], Bold}]]
```

General: Value of option PlotRange –> {{–0.005, dBmax}, {6.78, 25}} is not All, Full, Automatic, a positive machine number, or an appropriate list of range specifications.

General: Value of option PlotRange –> {{–0.005, dBmax}, {6.78, 25}} is not All, Full, Automatic, a positive machine number, or an appropriate list of range specifications.

Out[•]=




```
Export["E:\\Haim\\for SPP paper 2024-2025\\Al400_DensityMap_with_Kinematics.png",
 %, "PNG", ImageResolution → 900]
```

Energy spectrum

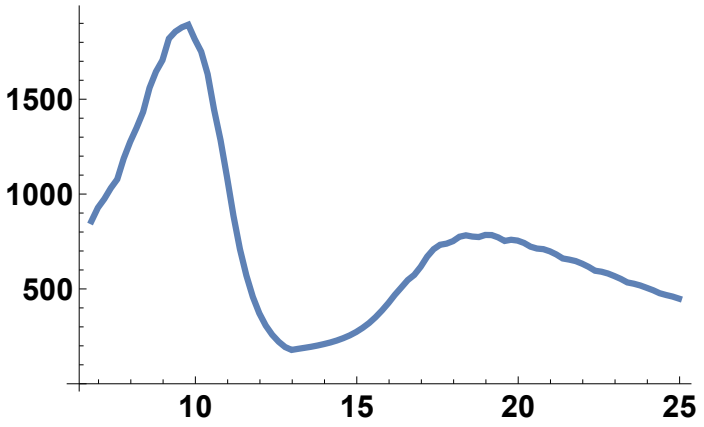
In[•]:=
```
(*Step 1:Flatten the data to make it easier to work with*)
flatData = Flatten[dataMap1, 1];

(*Step 2:Group data by each unique i value*)
groupedData = GatherBy[flatData, #[[2]] &];

(*Step 3:For each group (corresponding to each i),find the maximum intensity*)
maxIntensitiesForI = Table[{group[[1, 2]], Max[group[[All, 3]]]},
   (*Take the max intensity within each group of i*){group, groupedData}];

(*Step 4:Plot the maximum intensity for each i*)
ListLinePlot[maxIntensitiesForI, Joined → True, PlotRange → All,
 PlotStyle → Thickness[0.01], PlotLabel → None, LabelStyle → {16, GrayLevel[0], Bold}]
```

Out[•]=




```
Export["E:\\Haim\\for SPP paper 2024-2025\\Al400_Spectrum.png",
 %, "PNG", ImageResolution → 900]
```