Initialize plot formatting and set global variables

```
SetOptions[{ListPlot, ListLogPlot, ListLogLogPlot, LogLogPlot},
  Frame → True,
  Axes → False,
  FrameStyle → Directive[Black, 16, FontFamily → "Times New Roman"],
  PlotStyle → Black];
```

Define functions and constants

```
(*#####################*)
(*#### Constants ####*)
(*#####################*)
kB = 1.3806485 * 10^-23 (*Boltzmann's cst (J K^-1) *);
h = 6.6260696 * 10^-34 (* Plank's cst (J s)*);
hbar = h / (2 π) (*Plank's cst (J s)*);

(*##########################################################*)
(*### Scattering off of an array of linear defects. ####*)
(*##########################################################*)

qm[m_] := 2 π m / d (*d is D in paper*);

qxmσ[kx_, ky_, m_, sign_] := -kx + (sign) (kx^2 + 2 ky qm[m] - qm[m]^2)^(1/2);

(*Prefactor including forward scattering and Jacobian.*)
VpreSTGB[m_, kx_, ky_, sign_] :=
  (kx^2 - (sign) kx (kx^2 + 2 ky qm[m] - qm[m]^2 )^(1/2) + ky qm[m]) / (kx^2 + 2 ky qm[m] - qm[m]^2)^(1/2) // N;

ΓARRAY[kx_, ky_, kz_, V1Twidle2_] :=
 Module[{k = (kx^2 + ky^2 + kz^2)^(1/2), m, mmaxPos, mmaxNeg, sum = 0},
  mmaxPos = IntegerPart[d/(2 π) (ky + (kx^2 + ky^2)^(1/2))];
  mmaxNeg = Abs[IntegerPart[d/(2 π) (ky - (kx^2 + ky^2)^(1/2))]];
  (*Calculate m=0 term in the sum, the sum over σ=+- is built in.
     This term is present at all values of k.*)
  sum = sum + (VpreSTGB[0, kx, ky, 1] V1Twidle2[-2 kx, -qm[0], k] +
      VpreSTGB[0, kx, ky, -1] V1Twidle2[-2 kx, -qm[0], k]);

  (*Calculate all m>0 terms in the sum, the sum over σ=+- is built in here.
      This limits the number of terms in the sum over m.*)
```

```
  sum = sum + Sum[((VpreSTGB[m, kx, ky, 1] V1Twidle2[qxmσ[kx, ky, m, 1], -qm[m], k]) +
        (VpreSTGB[m, kx, ky, -1] V1Twidle2[qxmσ[kx, ky, m, -1], -qm[m], k])),
     {m, 1, mmaxPos, 1}];

  (*Calculate all m<0 terms in the sum,
  the sum over σ=+- is also built in here. Same criterion as above.*)
  sum = sum + Sum[
     ((VpreSTGB[-m, kx, ky, 1] V1Twidle2[qxmσ[kx, ky, -m, 1], -qm[-m], k]) +
        (VpreSTGB[-m, kx, ky, -1] V1Twidle2[qxmσ[kx, ky, -m, -1], -qm[-m], k])),
     {m, 1, mmaxNeg, 1}];
  (*Sums the terms calculated above and calculates a solution of τ^-1*)
  n1D / (d^2 k vg[k] hbar^2) sum
 ]

(*########################################
 ################################################*)
(*#### Functions for calculating phonon scattering off of GB strain fields,
Γgbs ####*)
(*########################################
 ################################################*)
(*|Ṽ1,Δ(qx,qy)|^2, hydorstatic strain field from a STGB*)
V1Twidle2εΔ[qx_, qy_, k_] := Abs[hbar ωk[k] γ (b (1 - 2 ν))/((1 - ν)) qy/(qx^2 + qy^2)]^2 // N;

(*|Ṽ1,S(qx,qy)|^2, pure shear strain field from a STGB*)
V1Twidle2εS[qx_, qy_, k_] := Abs[hbar ωk[k] γ b/((1 - ν)) (qx qy^2)/((qx^2 + qy^2)^2)]^2 // N;

(*|Ṽ1,R(qx,qy)|^2, rotational distortion about the z-axis from a STGB*)
V1Twidle2εR[qx_, qy_, k_] := Abs[hbar ωk[k] γ b (2 qx)/(qx^2 + qy^2)]^2 // N;
(*Sum scattering rate from hydrostatic, shear, and rotational strain components,
and convert to spherical coordinates.*)
ΓGBS[k_, θ_, ϕ_] :=
  (ΓARRAY[k Sin[θ] Cos[ϕ] // N, k Sin[θ] Sin[ϕ] // N, k Cos[θ] // N, V1Twidle2εΔ] +
    ΓARRAY[k Sin[θ] Cos[ϕ] // N, k Sin[θ] Sin[ϕ] // N, k Cos[θ] // N, V1Twidle2εS] +
    ΓARRAY[k Sin[θ] Cos[ϕ] // N, k Sin[θ] Sin[ϕ] // N, k Cos[θ] // N, V1Twidle2εR]);

(*Creates an array for the polar plots of τgbs*)
τPolarPlot[k_] := Module[{τϕlist},
   τϕlist = Table[{ΓGBS[k, π/2, ϕ]^-1 Cos[ϕ] * 10^9,
```

```
      ΓGBS[k, π/2, ϕ]^-1 Sin[ϕ] * 10^9}, {ϕ, 0, 2 π, 0.005}];
   ListPlot[τϕlist,
    FrameStyle → Directive[Black, 16, FontFamily → "Times New Roman"],
    PlotMarkers → {•, 5};
    PlotRange → {{-1.7, 1.7}, {-1, 1}},
    Joined → True,
    AspectRatio → 1,
    FrameLabel → {"τgbscos(ϕ) (ns)", "τgbssin(ϕ) (ns)"}]];

(*Numerically integrate τgbs[k,θ,ϕ] over θ and ϕ to get τgbs[k] which can be
 related to t[k]. (θ=0) and (ϕ=π/2) are omitted from the integral since the
 integrand is indeterminate at those locations. The (Sin[θ]Cos[ϕ])^2 are present
 because we are conserned with transport normal to the plane (kx direction). The
 addtional Sin[θ] is from converting to spherical coordinates.*)
τGBSθϕIntegrate[k_, n_] := Module[{integrand, θ, ϕ, Δ},
  Δ = (π/2.0)/n;
  integrand = 0;
  For[θ = Δ, θ ≤ π/2, θ = θ + Δ,
   For[ϕ = Δ, ϕ < π/2, ϕ = ϕ + Δ,
    integrand = integrand + Sin[θ - Δ/2]^3 Cos[ϕ - Δ/2]^2 / ΓGBS[k, θ - Δ/2, ϕ - Δ/2]]];
  8 * integrand * Δ * Δ * (3/(4 π))]

(*#########################################################*)
(*#### Calculate κL from a discrete list of values ####*)
(*#########################################################*)

(*Performing the θ and ϕ integration for τ^-1=Γgbs+τp-p^-1*)
τθϕIntegrate[k_, T_, n_] := Module[{integrand, θ, ϕ, Δ},
  Δ = (π/2.0)/n;
  integrand = 0;
  For[θ = Δ, θ ≤ π/2, θ = θ + Δ,
   For[ϕ = Δ, ϕ < π/2, ϕ = ϕ + Δ,
    integrand = integrand + Sin[θ - Δ/2]^3 Cos[ϕ - Δ/2]^2 / (ΓGBS[k, θ - Δ/2, ϕ - Δ/2] + τppModel[k, T, C1, C2]^-1)]];
```

```
8 * integrand * Δ * Δ * (3/(4 π))]

(*The total relaxation time, including phonon-phonon,
and phonon-GB scattering calculated via Matthiasen's rule.
  Calculating τTotal(ω) this way and plugging the result
  in to the isotropic Callaway model gives the same result as
  using Mattiessen's rule inside of the integral over θ and ϕ.*)
τTotCalc[τGBlist_, T_] := Table[{τGBlist[[i, 1]],
    (τppModel[ωk[τGBlist[[i, 1]]], T, C1, C2]^-1 + τPD[ωk[τGBlist[[i, 1]]], C3]^-1 +
       τGBlist[[i, 2]]^-1)^-1}, {i, 1, Length[τGBlist]}];
(*Kernal of the spectral heat capacity.*)
cvkern[T_, ωfunc_] := kB ((hbar ωfunc/(kB T))^2 Exp[hbar ωfunc/(kB T)]) / (Exp[hbar ωfunc/(kB T)] - 1)^2 ;

(*Calculate ϰL,xx,
where we have taken the single isotropic phonon mode approximation.
  Input an array for τtot and calculates lattice thermal conductivity.*)
ϰLCalc[τTotList_, T_] := Module[{integrand, i, Δk, ki},
  Δk = τTotList[[2, 1]] - τTotList[[1, 1]];
  integrand = 0;

  For[i = 1, i ≤ Length[τTotList], i++,
   ki = τTotList[[i, 1]];
   integrand = integrand + cvkern[T, ωk[ki]] * vg[ki]^2 * τTotList[[i, 2]] * ki^2];
  1/(2 π^2) integrand * Δk]


(*###########################################################*)
(*#### Semi-emperical forumlas for τgbs(ω) and t(ω) ####*)
(*###########################################################*)
τinvEmp[ω_, d_, b_, n1D_] :=
  8/3 n1D vs γ^2 (b/d)^2 +
   0.93 (1 + (1 - 2 ν)^2/(4 (1 - ν)^2) + 1/(32 (1 - ν)^2)) n1D γ^2 (b^2/d) (ω - 4 π vs/(3 d)) HeavisideTheta[ω - 4 π vs/(3 d)];

τinvEmpHigh[ω_, d_, b_, n1D_] := (1 + (1 - 2 ν)^2/(4 (1 - ν)^2) + 1/(32 (1 - ν)^2)) n1D γ^2 (b^2/d) (ω) ;

tEmp[ω_, d_, b_, n1D_] := (4 n1D vs τinvEmp[ω, d, b, n1D]^-1)/(3 + 4 n1D vs τinvEmp[ω, d, b, n1D]^-1);

tGen[ω_, α_] := (1 + α ω/ωD)^-1;
```

```
(*##########################################################################*)
(*#### Fuctions for calculating ϰL from semi-emperical expression ####*)
(*##########################################################################*)

(*#### calculate kmax from volume of PUC, vPUC ####*)
k0[vPUC_] := (6 π^2/vPUC)^(1/3);
(*#### BvK dispersion for transport modeling ####*)
ω0[vs_, vPUC_] := 2/π vs (6 π^2/vPUC)^(1/3);
ωkBvK[k_, ωmax_, kmax_] := ωmax Sin[π k/(2 kmax)];
kωBvK[ω_, ωmax_, kmax_] := (2 kmax)/π ArcSin[ω/ωmax]
  (*k as a function of ω in BvK*);
vgBvK[k_, ω0_, k0_] := (π ω0)/(2 k0) Cos[(k π)/(2 k0)];

(*#### Phonon-phonon scattering function. ####*)
τppModel[ω_, T_, B1_, B2_] := (B1 ω^2 T Exp[-B2/T])^-1
(*#### Point defect scattering function. #####*)
τPD[ω_, DPD_] := (DPD ω^4)^-1;

(*#### ϰL functions ####*)
Cs[ω_, kfunc_, vgfunc_, T_] := (3 hbar^2)/(2 π^2 kB T^2) (ω^2 kfunc^2 Exp[(hbar ω)/(kB T)])/(vgfunc (Exp[(hbar ω)/(kB T)] - 1)^2);
ϰL[T_, τfunc_, vgfunc_, kfunc_, intlimits_] :=
  1/3 NIntegrate[Cs[ω, kfunc, vgfunc, T] vgfunc^2 τfunc, intlimits];
ϰLω[ω_, T_, τfunc_, vgfunc_, kfunc_] := 1/3 Cs[ω, kfunc, vgfunc, T] vgfunc^2 τfunc;


(*###########################*)
(*#### Misc. Functions ####*)
(*###########################*)

(*#### Readies data for plotting ####*)
τGBExtract[τGBList_, n_] :=
  Table[{τGBList[[n, 3, i, 1]] vs/ωD, τGBList[[n, 3, i, 2]] * 10^9},
   {i, 1, Length[τGBList[[n, 3]]]}];
ΓGBExtract[τGBList_, n_] := Table[{τGBList[[n, 3, i, 1]] vs/ωD,
    (τGBList[[n, 3, i, 2]] * 10^9)^-1}, {i, 1, Length[τGBList[[n, 3]]]}];
```

```
tGBExtract[τGBList_, n_] := Table[{τGBList[[n, 3, i, 1]] vs / ωD,
     (4 n1D vs τGBList[[n, 3, i, 2]]) / (3 + 4 n1D vs τGBList[[n, 3, i, 2]])}, {i, 1, Length[τGBList[[n, 3]]]}];

(*#### Power law reference lines ####*)
T2[norm_, Trange_] := Table[{Tref, Tref^2 / Trange[[2]]^2 norm // N}, Trange];
T3[norm_, Trange_] := Table[{Tref, Tref^3 / Trange[[2]]^3 norm // N}, Trange];

(*#### Calculates the GB angle from the
 burgers vector and the dislocation spacing. ####*)
θcalc[b_, s_] := 2 ArcTan[(b / (2 s))] (180. / π);
```