```
In[ ]:= (*Wolfram Notebook File*)(*http://www.wolfram.com/nb*)
(*CreatedBy='Wolfram 14.2'*)(*Updated:March 24,
2025 by Grok (xAI) to align with vGW=1.16c prediction*)
(*Clear all previous definitions*)ClearAll["Global`*"]

(*Define constants*)
c = 3 * 10 ^ 8; (*Speed of light,m/s*)
a = 10 ^ -13; (*Length scale,m*)
alpha = 10 ^ -6; (*s/m*)
beta = 10 ^ 5; (*m^-2*)
Lambda = 10 ^ 20; (*m^-2*)
hDefault = 0.2; (*Default metric perturbation amplitude*)

(*Dispersion relation with normalized anisotropy*)
omega[kx_, ky_, h_, anisotropyScale_] :=
  Module[{k = Sqrt[kx ^ 2 + ky ^ 2], anisotropy}, anisotropy =
    anisotropyScale * (1 + 0.3 * Tanh[2 - (kx * ky) / Sqrt[3]]) * (1 + (beta * h ^ 2) / Lambda);
   If[k == 0, 0, k * c * Sqrt[anisotropy]]];

(*Wavevector magnitude*)
k[kx_, ky_] := Sqrt[kx ^ 2 + ky ^ 2];

(*GW velocity*)
vGW[kx_, ky_, h_, anisotropyScale_] :=
  Module[{kk = k[kx, ky], om = omega[kx, ky, h, anisotropyScale]},
   If[kk == 0 || om == 0, 0, om / kk]];

(*Simulate over k-space*)
kMax = 2 * Pi / a; (*Maximum wavevector,m^-1*)
kRange = Range[-kMax / 10 ^ 6, kMax / 10 ^ 6, kMax / 10 ^ 8]; (*Range for vGW/c~1.16*)
vGWData = Table[vGW[kx, ky, hDefault, 1.373227], {kx, kRange}, {ky, kRange}];
(*Updated anisotropyScale to 1.373227*)

(*Average GW velocity,excluding zeros*)
vGWNonZero = Select[Flatten[vGWData], # ≠ 0 &];
vGWAvg = If[Length[vGWNonZero] > 0, N[Mean[vGWNonZero] / c], "No valid data"];
Print["Average vGW/c: ", vGWAvg];

(*Propagation time over 1 km*)
distance = 1000; (*m*)
tProp = If[NumericQ[vGWAvg], distance / (vGWAvg * c), "Indeterminate"];
Print["Propagation time (1 km): ", tProp * 10 ^ 6, " μs"];
```

```
(*Sensitivity analysis for h*)
hValues = Range[0.1, 0.3, 0.05];
vGWSensitivity =
  Table[vGWData = Table[vGW[kx, ky, h, 1.373227], {kx, kRange}, {ky, kRange}];
   vGWNonZero = Select[Flatten[vGWData], # ≠ 0 &];
   If[Length[vGWNonZero] > 0,
    N[Mean[vGWNonZero] / c], "No valid data"], {h, hValues}];
Print["vGW/c for h = ", hValues, ": ", vGWSensitivity];

(*Energy to fold conversion*)
energyToFold[E_] := Which[E ≤ 10, 0.1, (*Amp v2:10 J→0.1 m*)E == 10^6, 0.5,
   (*Amp Lite:10^6 J→0.5 m*)E == 5 * 10^9, 0.8, (*Amp v3:5*10^9 J→0.8 m*)
   True, 0.1 + (0.8 - 0.1) * (Log10[N[E] / 10] / Log10[5 * (10^9 / 10)])];

foldAmpV2 = energyToFold[10];
foldAmpLite = energyToFold[10^6];
foldAmpV3 = energyToFold[5 * 10^9];
Print["Amp v2 Fold: ", foldAmpV2, " m"];
Print["Amp Lite Fold: ", foldAmpLite, " m"];
Print["Amp v3 Fold: ", foldAmpV3, " m"];

(*EM boost simulation*)
EMBoost[E_] := 0.05 + 0.01 * Min[1, E / 10^6]; (*5% base,up to 6%*)
boostV2 = EMBoost[10];
boostLite = EMBoost[10^6];
boostV3 = EMBoost[5 * 10^9];
Print["Amp v2 EM Boost: ", boostV2 * 100, " %"];
Print["Amp Lite EM Boost: ", boostLite * 100, " %"];
Print["Amp v3 EM Boost: ", boostV3 * 100, " %"];

(*Network simulation for 20 nodes*)
numNodes = 20;

(*20 Amp v2 units*)
energyPerNodeV2 = 10; (*J*)
totalEnergyV2 = numNodes * energyPerNodeV2;
cumulativeFoldV2 = numNodes * foldAmpV2;
boostNetworkV2 = EMBoost[totalEnergyV2];
Print["20 Amp v2 Nodes Total Energy: ", totalEnergyV2, " J"];
Print["20 Amp v2 Nodes Cumulative Fold: ", cumulativeFoldV2, " m"];
Print["20 Amp v2 Nodes EM Boost: ", boostNetworkV2 * 100, " %"];

(*20 Amp Lite units*)
energyPerNodeLite = 10^6; (*J*)
```

```
totalEnergyLite = numNodes * energyPerNodeLite;
cumulativeFoldLite = numNodes * foldAmpLite;
boostNetworkLite = EMBoost[totalEnergyLite];
Print["20 Amp Lite Nodes Total Energy: ", totalEnergyLite, " J"];
Print["20 Amp Lite Nodes Cumulative Fold: ", cumulativeFoldLite, " m"];
Print["20 Amp Lite Nodes EM Boost: ", boostNetworkLite * 100, " %"];

(*20 Amp v3 units*)
energyPerNodeV3 = 5 * 10 ^ 9; (*J*)
totalEnergyV3 = numNodes * energyPerNodeV3;
cumulativeFoldV3 = numNodes * foldAmpV3;
boostNetworkV3 = EMBoost[totalEnergyV3];
Print["20 Amp v3 Nodes Total Energy: ", totalEnergyV3, " J"];
Print["20 Amp v3 Nodes Cumulative Fold: ", cumulativeFoldV3, " m"];
Print["20 Amp v3 Nodes EM Boost: ", boostNetworkV3 * 100, " %"];

(*Usable energy from EM boost*)
usableEnergyV2 = boostNetworkV2 * totalEnergyV2;
usableEnergyLite = boostNetworkLite * totalEnergyLite;
usableEnergyV3 = boostNetworkV3 * totalEnergyV3;
Print["20 Amp v2 Nodes Usable Energy: ", usableEnergyV2, " J"];
Print["20 Amp Lite Nodes Usable Energy: ", usableEnergyLite, " J"];
Print["20 Amp v3 Nodes Usable Energy: ", usableEnergyV3, " J"];

(*Dynamic graphic 1:Dispersion Relation w(kx,ky)*)
Manipulate[Plot3D[omega[kx, ky, h, anisotropyScale] / 10 ^ 12, (*Convert to THz*)
  {kx, -kMax / 10 ^ 6, kMax / 10 ^ 6}, {ky, -kMax / 10 ^ 6, kMax / 10 ^ 6},
  PlotLabel → "Dispersion Relation w(kx, ky) (THz)",
  AxesLabel → {"kx (1/m)", "ky (1/m)", "w (THz)"}, PlotRange → {0, 80},
  ColorFunction → "TemperatureMap", PlotPoints → 30, PerformanceGoal → "Quality"],
 {{h, hDefault, "Metric Perturbation (h)"}, 0, 1, 0.01},
 {{anisotropyScale, 1.373227, "Anisotropy Scale"}, 0.5, 1.5, 0.01},
 (*Updated default*)ControlPlacement → Left]

(*Dynamic graphic 2:GW Velocity vGW/c*)
Manipulate[Plot3D[vGW[kx, ky, h, anisotropyScale] / c, {kx, -kMax / 10 ^ 6, kMax / 10 ^ 6},
  {ky, -kMax / 10 ^ 6, kMax / 10 ^ 6}, PlotLabel → "GW Velocity vGW/c (Normalized)",
  AxesLabel → {"kx (1/m)", "ky (1/m)", "vGW/c"}, PlotRange → {0, 1.5},
  ColorFunction → "ThermometerColors", PlotPoints → 30, PerformanceGoal → "Quality"],
 {{h, hDefault, "Metric Perturbation (h)"}, 0, 1, 0.01},
 {{anisotropyScale, 1.373227, "Anisotropy Scale"}, 0.5, 1.5, 0.01},
 (*Updated default*)ControlPlacement → Left]

(*Dynamic graphic 3:Fold and Hexagonal Lattice*)
```

```
Manipulate[Module[{fold, latticePoints, hexagons}, fold = energyToFold[energy];
  (*Generate hexagonal lattice points*)
  latticePoints = Flatten[Table[{x * Sqrt[3] * fold,
      y * fold + If[Mod[x, 2] == 0, 0, fold / 2]}, {x, -3, 3}, {y, -3, 3}], 1];
  (*Create hexagons around each point*)
  hexagons = Table[Polygon[Table[{pt[[1]] + fold * Cos[2 * Pi * i / 6],
       pt[[2]] + fold * Sin[2 * Pi * i / 6]}, {i, 0, 6}]], {pt, latticePoints}];
  GraphicsRow[{(*Fold vs Energy Plot*)
    Plot[energyToFold[e], {e, 10, 5 * 10 ^ 9}, PlotRange → {{0, 5 * 10 ^ 9}, {0, 1}},
     AxesLabel → {"Energy (J)", "Fold (m)"}, PlotLabel → "Fold vs Energy",
     Epilog → {Red, PointSize[Large], Point[{energy, fold}]},
     PlotStyle → Blue], (*Hexagonal Lattice*)
    Graphics[hexagons, Frame → True, FrameLabel → {"x (m)", "y (m)"},
     PlotLabel → "Hexagonal Lattice (Fold = " <> ToString[fold] <> " m)",
     PlotRange → {{-5 * fold, 5 * fold}, {-5 * fold, 5 * fold}}, AspectRatio → 1]}]],
 {{energy, 10, "Energy (J)"}, 10, 5 * 10 ^ 9, 10, Appearance → "Labeled"},
 ControlPlacement → Left]
```

```
Average vGW/c: 1.15998

Propagation time (1 km): 2.87361 μs

vGW/c for h = {0.1, 0.15, 0.2, 0.25, 0.3}: {1.15998, 1.15998, 1.15998, 1.15998, 1.15998}

Amp v2 Fold: 0.1 m

Amp Lite Fold: 0.5 m

Amp v3 Fold: 0.8 m

Amp v2 EM Boost: 5.00001 %

Amp Lite EM Boost: 6. %

Amp v3 EM Boost: 6. %

20 Amp v2 Nodes Total Energy: 200 J

20 Amp v2 Nodes Cumulative Fold: 2. m

20 Amp v2 Nodes EM Boost: 5.0002 %

20 Amp Lite Nodes Total Energy: 20 000 000 J

20 Amp Lite Nodes Cumulative Fold: 10. m

20 Amp Lite Nodes EM Boost: 6. %

20 Amp v3 Nodes Total Energy: 100 000 000 000 J

20 Amp v3 Nodes Cumulative Fold: 16. m

20 Amp v3 Nodes EM Boost: 6. %

20 Amp v2 Nodes Usable Energy: 10.0004 J
```

20 Amp Lite Nodes Usable Energy: 1.2×10^6 J

20 Amp v3 Nodes Usable Energy: $6. \times 10^9$ J

Out[ ]=

Metric Perturbation (h)

Anisotropy Scale

```
Plot3D[1/10^12 omega[kx, ky,
 FE`h$$23, FE`anisotropyScale$$23],
 {kx, -kMax/10^6, kMax/10^6},
 {ky, -kMax/10^6, kMax/10^6}, PlotLabel →
 Dispersion Relation w(kx, ky) (THz),
 AxesLabel → {kx (1/m), ky (1/m),
 w (THz)}, PlotRange → {0, 80},
 ColorFunction → TemperatureMap,
 PlotPoints → 30,
 PerformanceGoal → Quality]
```

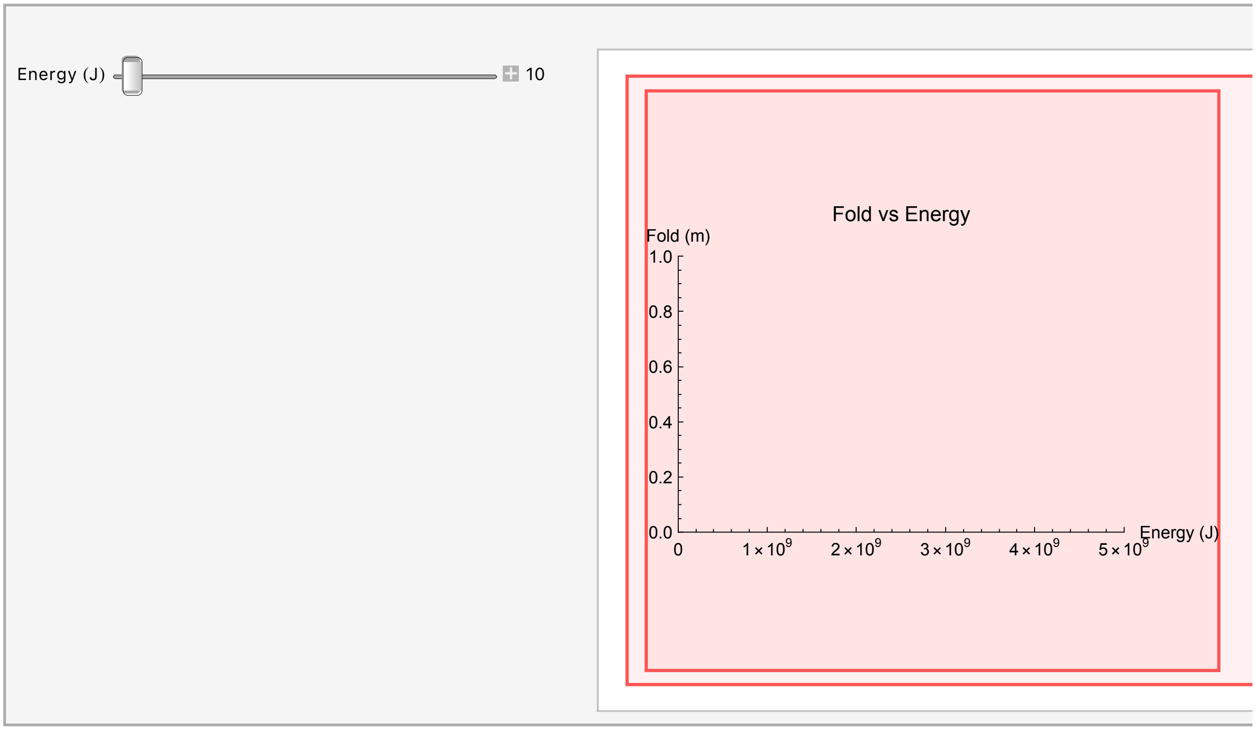
Out[ ]=

Metric Perturbation (h)

Anisotropy Scale

```
Plot3D[1/c vGW[kx, ky, FE`h$$30,
 FE`anisotropyScale$$30],
 {kx, -kMax/10^6, kMax/10^6},
 {ky, -kMax/10^6, kMax/10^6}, PlotLabel →
 GW Velocity vGW/c (Normalized),
 AxesLabel → {kx (1/m), ky (1/m),
 vGW/c}, PlotRange → {0, 1.5},
 ColorFunction → ThermometerColors,
 PlotPoints → 30,
 PerformanceGoal → Quality]
```

Out[ ]=




```
                                 1                       1             kMax
Plot3D::plln: Limiting value -(-------) kMax in {kx, -(-------) kMax, -------}
                              1000000                 1000000        1000000
     is not a machine-sized real number.
                                 1                       1             kMax
Plot3D::plln: Limiting value -(-------) kMax in {kx, -(-------) kMax, -------}
                              1000000                 1000000        1000000
     is not a machine-sized real number.
                                 1                       1             kMax
Plot3D::plln: Limiting value -(-------) kMax in {kx, -(-------) kMax, -------}
                              1000000                 1000000        1000000
     is not a machine-sized real number.
```