Paschen Curve Monte Carlo Simulations

William Bowers
Eastside Preparatory School
Kirkland, WA
In collaboration with Sophia Gershaft, Dr. Charles Whitmer, and Gunnar Mein

Basic 1D parallel plate simulations

In[•]:=
```
AvgElectronsBanked[Nc_, Ni_, count_] := Module[{result, stack},
stack = CreateDataStructure["Stack"];
result = Table[
Module[{pos1, pos2, numElectrons, lambdaI},
numElectrons = 1;
stack["Push", 0];
While[! stack["EmptyQ"],
pos1 = stack["Pop"];
While[(pos2 = pos1 + RandomVariate[ExponentialDistribution[1]]) ≤ Nc,
lambdaI = Nc / Ni;
If[(pos2 - pos1) ≥ lambdaI, numElectrons++;
        stack["Push", pos2] ];
Sow[pos2];
pos1 = pos2];];
numElectrons],
{count}];
Around[Mean[result], StandardDeviation[result] / Sqrt[count]]]
```

```
In[ ]:= AvgElectronsWeighted[Nc_, Ni_, count_] := Module[{result},
result = Table[
Module[{pos1, pos2, numElectrons, lambdaI, weight},
numElectrons = 1;
pos1 = 0;
weight = 1;
While[(pos2 = pos1 + RandomVariate[ExponentialDistribution[1]]) ≤ Nc,
lambdaI = Nc / Ni;
If[(pos2 - pos1) ≥ lambdaI, numElectrons += weight;
        weight *= 2];
pos1 = pos2];
numElectrons],
{count}];
Around[Mean[result], StandardDeviation[result] / Sqrt[count]]]
```

Defining 3D MC utility functions

```
In[ ]:= GetTime[s_, vx_, vw_, ax_] := Module[{S, set, start},
If[vw > 0,
S[t_] := 1 / (2 ax) * ((vx + ax t) Sqrt[vw^2 + (vx + ax t)^2] -
        vx Sqrt[vw^2 + vx^2] + vw^2 (ArcSinh[(ax t + vx) / vw] - ArcSinh[vx / vw])),
S[t_] := (-vx Sqrt[vx^2] + (ax t + vx) Sqrt[(ax t + vx)^2]) / (2 ax)
];
set = FindRoot[S[t] == s, {t, 0.05, 10}];
t /. set]
```

```
In[ ]:= GetVelocity[energy_, cosθ_, ϕ_] := Module[{q, m, e0, v, vx1, vy1, vz1},
q = 1;
m = 511 000;
v = Sqrt[2 energy / m];
vx1 = v Cos[ϕ] Sqrt[1 - cosθ^2];
vy1 = v Sin[ϕ] Sqrt[1 - cosθ^2];
vz1 = v cosθ;
{vx1, vy1, vz1}]
```

```
In[ ]:= GetNewPositions[E_, s_, x0_, y0_, z0_, vx_, vy_, vz_] :=
 Module[{vw, q, m, ax, set, x, y, z, t, velocities},
vw = Sqrt[vy^2 + vz^2];
q = 1;
m = 511 000;
ax = (q E) / m;
t = GetTime[s, vx, vw, ax];
x =  vx t + 1 / 2 ax t^2;
y = vy t;
z = vz t;
{x0 + x, y0 + y, z0 + z, vx + ax t, vy, vz, 1 / 2 * m * ((vx + ax t)^2 + vy^2 + vz^2)}
 ]

In[ ]:= UpdateVelocity[energy_, vx_, vy_, vz_, tbound_] :=
 Module[{m, cost, phi, a, pvec1, pvec2, pvec3, b, c, v, res},
  If[energy == 0, Return[{0, 0, 0, 100, 100}]];
  m = 511 000;
  cost = RandomReal[{tbound, 1}, WorkingPrecision → 16];
  phi = RandomReal[2 Pi, WorkingPrecision → 16];
  a = Normalize[{vx, vy, vz}];
  pvec1 = Normalize[{-vy, -vx, -0}];
  pvec2 = Normalize[{0, -vz, vy}];
  pvec3 = Normalize[{-vz, 0, vx}];
  b = pvec1;
  If[Abs[Dot[a, pvec2]] < Abs[Dot[a, b]], b = pvec2];
  If[Abs[Dot[a, pvec3]] < Abs[Dot[a, b]], b = pvec3];
  c = Cross[a, b];
  v = Sqrt[2 energy / m];
  res = SetPrecision[
    v (a cost + b Sqrt[1 - cost^2] Cos[phi] + c Sin[phi] Sqrt[1 - cost^2]), 16];
  Flatten[Append[res, {cost, phi}]]
 ]
```

In[•]:=
```
GetNewPositionsEuler[Δt_, s_, V_, x0_, y0_, z0_, vx0_, vy0_, vz0_] :=
 Module[{q, m, path, eField, R, x, y, z, vx, vy,
   vz, ex, ey, ez, e0, e1, Δpos, Δr, Δtsum, scalefactor},
  q = 1;
  m = 511 000;
  {x, y, z} = {x0, y0, z0};
  {vx, vy, vz} = {vx0, vy0, vz0};
  path = 0;
  Δtsum = 0;
  While[path < s,
   (*check energy conservation with delta t - take rms error *)
   (*Sow[{Δtsum,x,y,z,vx,vy,vz,ez,ey,ez}];*)
   Δtsum += Δt;
   Δpos = Δt {vx, vy , vz};
   Δr = Norm[Δpos];
   If[path + Δr > s ,
    scalefactor = (s - path) / Δr;
    Δpos *= scalefactor;
    {x, y, z} += Δpos;
    path += Norm[Δpos];
    If[! InBounds[x, y, z], Return[{x, y, z, vx, vy, vz, 0, 0, 0, 0}]];
    {ex, ey, ez} = GetEField[V, x, y, z];
    {vx, vy, vz} += ((q (Δt * scalefactor)) / m) {ex, ey, ez};
    Break[];
   ];
   {x, y, z} += Δpos;
   path += Norm[Δpos];
   If[! InBounds[x, y, z], Return[{x, y, z, vx, vy, vz, 0, 0, 0, 0}]];
   {ex, ey, ez} = GetEField[V, x, y, z];
   {vx, vy, vz} += ((q Δt) / m) {ex, ey, ez};
  ];
  (*Sow[{Δtsum,x,y,z,vx,vy,vz}];*)
  e0 = 1 / 2 * m * (vx0^2 + vy0^2 + vz0^2) +
    GetPotential[V, x0, y0, z0] - GetPotential[V, x, y, z];
  e1 = 1 / 2 * m * (vx^2 + vy^2 + vz^2);
  {x, y, z, vx, vy, vz, e1, e1 - e0, path, s}]
```

Defining utility functions for electrode geometries

In[•]:=
```
(*Parallel plate*)
d = 5;
ID = "PP";
GetStartingPos[] := {0, 0, 0, 0, 0, 0};
InBounds[x_, y_, z_] := 0 ≤ x ≤ d;
GetEField[V_, x_, y_, z_] := {V / d, 0, 0};
GetPotential[V_, x_, y_, z_] := V x / d;
```

In[•]:=
```
(*Concentric spheres*)
a = 2;
b = 7;
ID = "SS";
GetStartingPos[] := Flatten[
   Append[RandomPoint[Sphere[{0, 0, 0}, a], 1, WorkingPrecision → 16], {0, 0, 0}]];
InBounds[x_, y_, z_] := a ≤ Norm[{x, y, z}] ≤ b
(*k/r^2 → k = Vab/(b-a)*)
GetEField[V_, x_, y_, z_] :=
  {x, y, z } (V a b) / (Norm[SetPrecision[{x, y, z} // N, 16]]^3 (b - a));
(*GetEField[V_,x_,y_,z_] :={x,y,z } (V a b)/(Norm[{x,y,z}]^3 (b-a)); *)
GetPotential[V_, x_, y_, z_] := (V a b) / (Norm[{x, y, z}] (b - a));
```

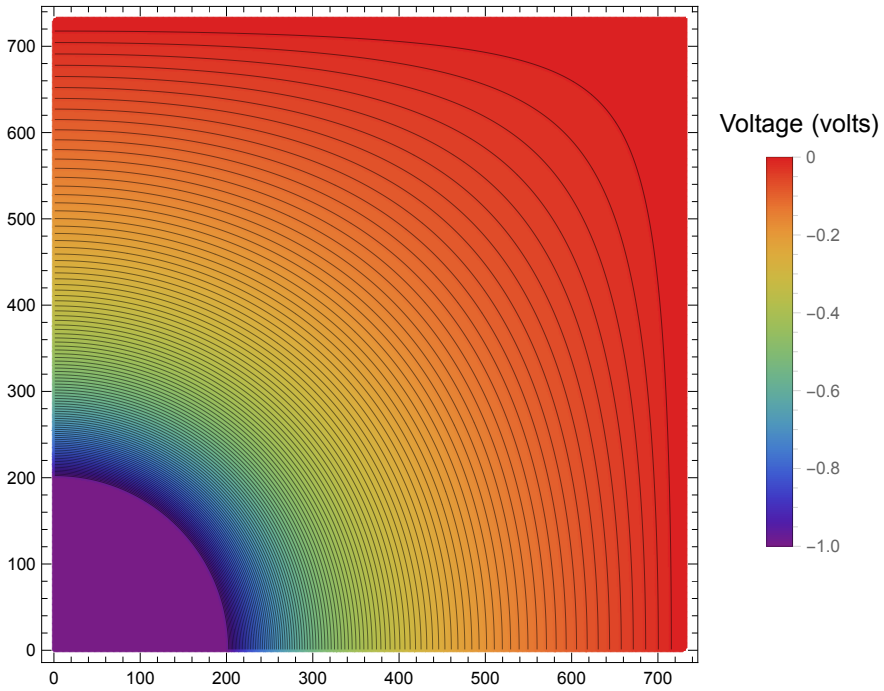
In[•]:=
```
(*Concentric spheres discrete*)
sspotential = Import["/Users/wbowers/Downloads/phi2.txt", "Table"];
sspotential = Drop[sspotential, 2];
a = 2;
b = 7;
ID = "SS Discrete";
GetStartingPos[] := Flatten[
   Append[RandomPoint[Sphere[{0, 0, 0}, a], 1, WorkingPrecision → 16], {0, 0, 0}]];
InBounds[x_, y_, z_] := a ≤ Norm[{x, y, z}] ≤ b
gs = 0.01;
GetPotential[V_, x_, y_, z_] := Module[{ρ, length, i, j, α, β},
  ρ = Sqrt[x^2 + y^2];
  i = Floor[ρ / gs];
  j = Floor[Abs[z] / gs];
  length = Length[sspotential];
  If[i + 2 > length || j + 2 > length, Return[0.]];
  α = ρ / gs - i;
  β = Abs[z] / gs - j;
  -V ((1 - β) (1 - α) sspotential⟦i + 1, j + 1⟧ + (1 - β) α sspotential⟦i + 2, j + 1⟧ +
     β (1 - α) sspotential⟦i + 1, j + 2⟧ + β α sspotential⟦i + 2, j + 2⟧)
 ]
GetEField[V_, x_, y_, zraw_] :=
 Module[{z, ρ, i, j, length, α, β, Er, Ez, Eρ, Ezf, phi, flip},
  ρ = Sqrt[x^2 + y^2];
  i = Floor[ρ / gs];
  If[zraw < 0, z = Abs[zraw]; flip = True, z = zraw];
  j = Floor[z / gs];
  length = Length[sspotential];
  If[i + 2 > length || j + 2 > length, Return[{0., 0., 0.}]];
  α = ρ / gs - i;
  β = z / gs - j;
  Eρ = (1 - β) (sspotential⟦i + 1, j + 1⟧ - sspotential⟦i + 2, j + 1⟧) / gs +
    β (sspotential⟦i + 1, j + 2⟧ - sspotential⟦i + 2, j + 2⟧) / gs;
  Ezf = (1 - α) (sspotential⟦i + 1, j + 1⟧ - sspotential⟦i + 1, j + 2⟧) / gs +
    α (sspotential⟦i + 2, j + 1⟧ - sspotential⟦i + 2, j + 2⟧) / gs;
  If[flip, Ezf = -Ezf];
  -V Flatten[{Eρ * Normalize[{x, y}], Ezf}]
 ]
```

```
GetPotentialContinuous[V_, x_, y_, z_] := Module[{ρ, length, i, j, α, β},
  ρ = Sqrt[x^2 + y^2];
  i = Floor[ρ / gs];
  j = Floor[Abs[z] / gs];
  length = Length[sspotential];
  If[i + 2 > length || j + 2 > length, Return[0.]];
  α = ρ / gs - i;
  β = Abs[z] / gs - j;
  -V ((1 - β) (1 - α) sspotential⟦i + 1, j + 1⟧ + (1 - β) α sspotential⟦i + 2, j + 1⟧ +
     β (1 - α) sspotential⟦i + 1, j + 2⟧ + β α sspotential⟦i + 2, j + 2⟧)
 ]
GetEFieldContinuous[V_, x_, y_, zraw_] :=
 Module[{z, ρ, i, j, length, α, β, Er, Ez, Eρ, Ezf, phi, flip},
  ρ = Sqrt[x^2 + y^2];
  i = Floor[ρ / gs];
  If[zraw < 0, z = Abs[zraw]; flip = True, z = zraw];
  j = Floor[z / gs];
  length = Length[sspotential];
  If[i + 2 > length || j + 2 > length, Return[{0., 0., 0.}]];
  α = ρ / gs - i;
  β = z / gs - j;
  Eρ = (1 - β) (sspotential⟦i + 1, j + 1⟧ - sspotential⟦i + 2, j + 1⟧) / gs +
    β (sspotential⟦i + 1, j + 2⟧ - sspotential⟦i + 2, j + 2⟧) / gs;
  Ezf = (1 - α) (sspotential⟦i + 1, j + 1⟧ - sspotential⟦i + 1, j + 2⟧) / gs +
    α (sspotential⟦i + 2, j + 1⟧ - sspotential⟦i + 2, j + 2⟧) / gs;
  If[flip, Ezf = -Ezf];
  -V Flatten[{Eρ * Normalize[{x, y}], Ezf}]
 ]
```

In[•]:=
```
(*Sphere in cylinder*)
scpotential = Import["/Users/wbowers/Downloads/new-cwPhi.txt", "Table"];
GetScPotential[i_, j_] :=
  If[i > Length[scpotential] || j > Length[scpotential], 0, scpotential[[i, j]]];
a = 2;
b = 7.3;
h = 14.6;
ID = "SC";
lmax = 10;
binwidth = 0.05;
gs = 100 * scpotential[[3, 3]];
phiFile = "Phi file: " <> StringDelete[ToString[scpotential[[1]]], {",", "{", "}"}];
scpotential = Drop[scpotential, 3];
GetAngles[x_, y_, z_] := {z / Sqrt[x^2 + y^2 + z^2], ArcTan[x, y]};
FindBinNum[costheta_, binwidth_] := Floor[(costheta + 1) / binwidth] + 1;
GetStartingPos[] := Flatten[
   Append[RandomPoint[Sphere[{0, 0, 0}, a], 1, WorkingPrecision → 16], {0, 0, 0}]];
InBounds[x_, y_, z_] :=
  a ≤ Sqrt[x^2 + y^2 + z^2] && Sqrt[x^2 + y^2] ≤ b && -h / 2 ≤ z ≤ h / 2;
GetPotential[V_, x_, y_, z_] := Module[{ρ, length, i, j, α, β},
  ρ = Sqrt[x^2 + y^2];
  i = Floor[Abs[z] / gs];
  j = Floor[ρ / gs];
  α = Abs[z] / gs - i;
  β = ρ / gs - j;
  -V ((1 - β) (1 - α) GetScPotential[i + 1, j + 1] + (1 - β) α GetScPotential[i + 2, j + 1] +
      β (1 - α) GetScPotential[i + 1, j + 2] + β α GetScPotential[i + 2, j + 2])
 ]
GetEField[V_, x_, y_, zraw_] :=
 Module[{z, ρ, i, j, length, α, β, Er, Ez, Eρ, Ezf, phi, flip},
  ρ = Sqrt[x^2 + y^2];
  If[zraw < 0, z = Abs[zraw]; flip = True, z = zraw];
  i = Floor[z / gs];
  j = Floor[ρ / gs];
  α = z / gs - i;
  β = ρ / gs - j;
  Eρ = (1 - α) (GetScPotential[i + 1, j + 1] - GetScPotential[i + 1, j + 2]) / gs +
    α (GetScPotential[i + 2, j + 1] - GetScPotential[i + 2, j + 2]) / gs;
  Ezf = (1 - β) (GetScPotential[i + 1, j + 1] - GetScPotential[i + 2, j + 1]) / gs +
    β (GetScPotential[i + 1, j + 2] - GetScPotential[i + 2, j + 2]) / gs;
  If[flip, Ezf = -Ezf];
  -V Flatten[{Eρ * Normalize[{x, y}], Ezf}]
 ]
```

In[ ]:=
```
fig5 = ListContourPlot[scpotential, Contours → 100,
  PlotLegends → Placed[BarLegend[{"Rainbow", {-1, 0}},
    LegendLabel → "Voltage (volts)"], Automatic], ColorFunction → "Rainbow"]
```

Out[ ]=



In[ ]:=
```
Export["/Users/wbowers/Documents/stem-fellowship/tif-figs/e-field-contour.TIFF",
 fig5, ImageResolution → 600]
```

Out[ ]= /Users/wbowers/Documents/stem-fellowship/tif-figs/e-field-contour.TIFF

Creating e-field export for sphere-in-sphere

In[ ]:=
```
ssregion = RegionDifference[Ball[{0, 0, 0}, 7], Ball[{0, 0, 0}, 2]];
```

In[ ]:=
```
ssregion = RegionDifference[Ball[{0, 0, 0}, 7], Ball[{0, 0, 0}, 2]];
Timing[spherepotential =
  NDSolveValue[{lapl, DirichletCondition[f[x, y, z] == -1, x^2 + y^2 + z^2 < 9],
    DirichletCondition[f[x, y, z] == 0, x^2 + y^2 + z^2 ≥ 9]}, f, {x, y, z} ∈ ssregion]]

sspotential = Table[Module[{val},
    val = SetPrecision[spherepotential[r, 0, z], 4];
    If[val === Indeterminate && Sqrt[r^2 + z^2] ≤ 2, val = -1];
    If[val === Indeterminate && Sqrt[r^2 + z^2] ≥ 6, val = 0];
    SetPrecision[val, 16]
   ], {z, 0, 7, 0.008}, {r, 0, 7, 0.008}];
```

In[ ]:=
```
sspotential = Insert[sspotential,
   "12/20/2022, William Bowers, a 1/r sphere-in-sphere potential", 1];
sspotential = Insert[sspotential, "701, 701, 0.0001", 2];
```

```
In[ ]:= Export["/Users/wbowers/Downloads/phi2.txt", SetPrecision[sspotential, 4], "Table"]
Out[ ]= /Users/wbowers/Downloads/phi2.txt
```

The main MC simulation (for parallel plate and sphere-in-sphere)

```
In[ ]:= AvgElectrons3D[Nc_, Ni_, D_, Ui_, tbound_, count_] :=
 Module[{V, E, λ, λi, result, stack},
  λ = D / Nc;
  λi = D / Ni;
  V = Ui * Ni;
  E = V / D;
  stack = CreateDataStructure["Stack"];
  result = Table[
    Module[{x0, x1, y0, y1, z0, z1, numElectrons, cosθ, ϕ, s, vx, vy, vz, energy},
     numElectrons = 1;
     stack["Push", {0, 0, 0, 0, 0, 0}];
     While[! stack["EmptyQ"],
      {x0, y0, z0, vx, vy, vz} = stack["Pop"];
      cosθ = 1;
      ϕ = 0;
      While[
       s = RandomVariate[ExponentialDistribution[1 / λ]];
       {x1, y1, z1, vx, vy, vz, energy} =
        GetNewPositions[E, s, x0, y0, z0, vx, vy, vz];
       0 ≤ x1 ≤ D,
       If[energy ≥ Ui,
        numElectrons++;
        stack["Push", {x1, y1, z1, 0, 0, 0}];
        energy -= Ui,
        (*else:*)
        energy -= Ui * RandomVariate[UniformDistribution[{0.1, 0.5}]];
        If[energy < 0, energy = 0];
       ];
       {vx, vy, vz} = UpdateVelocity[energy, vx, vy, vz, tbound];
       {x0, y0, z0} = {x1, y1, z1};
      ];
     ];
     numElectrons],
    {count}];

  Around[Mean[result], StandardDeviation[result] / Sqrt[count]]]

In[ ]:= AvgElectronsEuler[Nc_, Ni_, D_, Ui_, dt_, tbound_, print_, count_] :=
 Module[{eErrorList, V, E, λ, λi, result, stack, collisions, paths, svals, countbin,
```

```
 electronsbin, squaredbin, lbin, lbinSquared, data, seedcount, totalpaths},
λ = D / Nc;
λi = D / Ni;
V = Ui * Ni;
totalpaths = 0;
paths = {};
svals = 0;
E = V / D;
seedcount = 1;
collisions = {};
eErrorList = {};
stack = CreateDataStructure["Stack"];
result = Table[
  Module[{x0, x1, y0, y1, z0, z1, numElectrons, cosθ, ϕ,
    s, vx, vy, vz, energy, eError, numCollisions, pathval, sval, idx,
    cθ0, ϕ0, SH, l, l2, collisionE, cost, phi, start, path, input},
   numElectrons = 1;
   numCollisions = 0;
   stack["Push", GetStartingPos[]];
   {cθ0, ϕ0} = GetAngles[x0, y0, z0];
   While[! stack["EmptyQ"],
    {x0, y0, z0, vx, vy, vz} = stack["Pop"];
    cosθ = 1;
    ϕ = 0;
    energy = 0;
    path = 0;
    While[
     s = RandomVariate[ExponentialDistribution[1 / λ], WorkingPrecision → 16];
     {x1, y1, z1, vx, vy, vz, energy, eError, pathval, sval} =
      GetNewPositionsEuler[dt, s, V, x0, y0, z0, vx, vy, vz];
     AppendTo[eErrorList, eError];
     InBounds[x1, y1, z1],
     path += pathval;
     AppendTo[paths, pathval];
     svals += s;
     numCollisions++;
     If[energy ≥ Ui,
      numElectrons++;
      stack["Push", {x1, y1, z1, 0, 0, 0}];
      energy -= Ui,
      collisionE =
       SetPrecision[Ui * RandomVariate[UniformDistribution[{1 / 10, 1 / 2}]], 16];
      energy -= collisionE;
      If[energy < 0, energy = 0];
```

```
        ];
        {vx, vy, vz, cost, phi} = UpdateVelocity[energy, vx, vy, vz, tbound];
        {x0, y0, z0} = {x1, y1, z1};
       ];
       totalpaths += path;
      ];
      AppendTo[collisions, numCollisions];
      numElectrons],
     {count}];
   If[print,
    Print["1000 * energy error error: ", 1000 * RootMeanSquare[eErrorList]];
    Print["V/NC: ", V / Nc];
    Print["λ: ", λ];
    Print["Geometry: ", ID];
    Print["Collisions: ",
     Around[Mean[collisions], StandardDeviation[collisions] / Sqrt[count]]];
    Print["Mean path: ",  totalpaths / count / λ];
   ];
   Around[Mean[result], StandardDeviation[result] / Sqrt[count]]]
```

Exploring discrete sphere-in-cylinder E field

In[ ]:=
```
lpc = D[ϕ[r, z], z, z] + D[ϕ[r, z], r, r]  + 1 / r D[ϕ[r, z], r] ;
TraditionalForm[lpc]
```

Out[ ]//TraditionalForm=

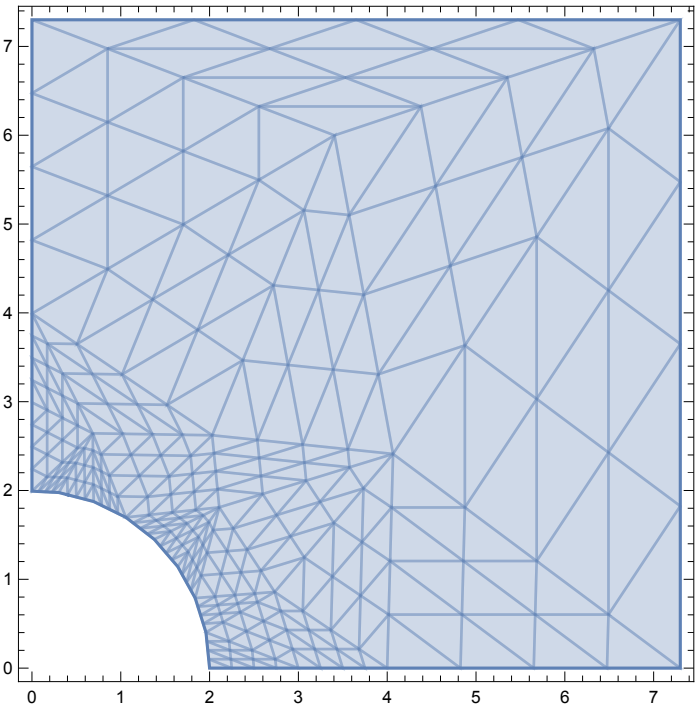
$$\phi^{(0,2)}(r, z) + \frac{\phi^{(1,0)}(r, z)}{r} + \phi^{(2,0)}(r, z)$$

In[ ]:=
```
region = RegionDifference[Rectangle[{0, 0}, {b, h / 2}], Disk[{0, 0}, a]]
```

Out[ ]=
```
BooleanRegion[#1 && ! #2 &, {Rectangle[{0, 0}, {7.3, 7.3}], Disk[{0, 0}, 2]}]
```

In[ ]:=
```
RegionPlot[region]
```

Out[ ]=



In[ ]:=
```
<< NDSolve`FEM`
```

In[ ]:=
```
mesh = ToElementMesh[region, MaxCellMeasure → 0.001]
```

Out[ ]=
```
ElementMesh[{{0., 7.3}, {0., 7.3}}, {TriangleElement[<79 083>]}]
```

In[ ]:=
```
Clear[z, r]
solc = NDSolveValue[
  {lpc == NeumannValue[0, r == 0 && a < z < b] + NeumannValue[0, z == 0 && a < r < b],
   DirichletCondition[ϕ[r, z] == -1, r^2 + z^2 == a^2],
   ϕ[r, h / 2] == 0, ϕ[b, z] == 0}, ϕ, {r, z} ∈ mesh
 ]
```

Out[ ]=
```
InterpolatingFunction[
```

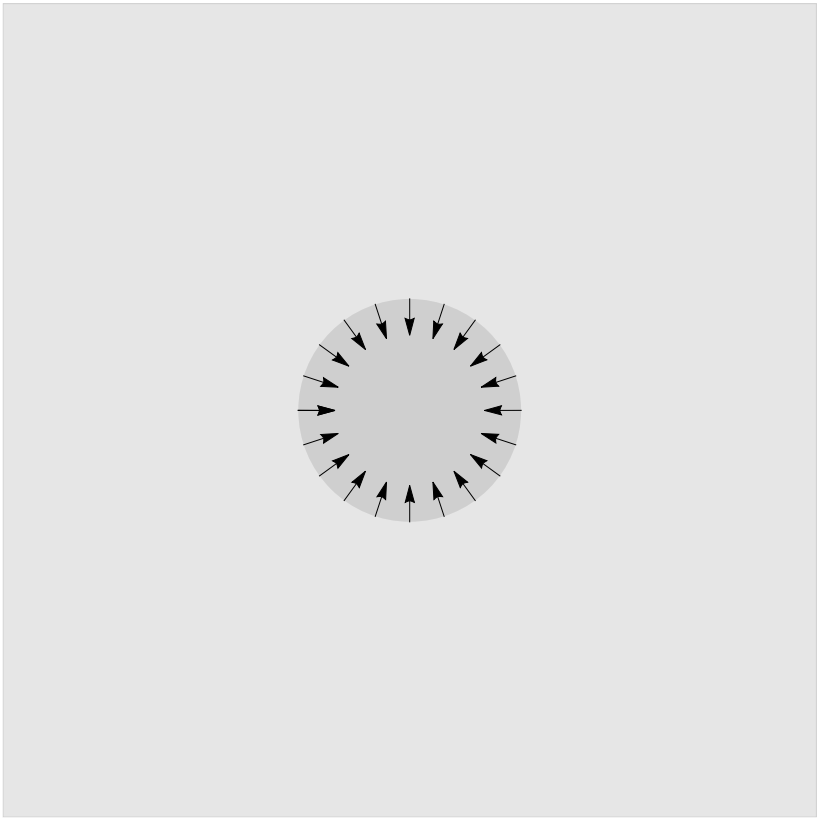

In[ ]:=
```
solc["ElementMesh"]
```

Out[ ]=
```
ElementMesh[{{0., 7.3}, {0., 7.3}}, {TriangleElement[<79 083>]}]
```

In[ ]:=
```
GetFromSolc[x_, z_] := If[Sqrt[x^2 + z^2] ≤ a, -1, solc[x, z]]
EstimateEField[x_, z_] := Module[{step, Er, Ez},
  step = 0.0005;
  Er =
   -(GetFromSolc[Abs[x + step / 2], Abs[z]] - GetFromSolc[Abs[x - step / 2], Abs[z]]) /
    step;
  Ez =
   -(GetFromSolc[Abs[x], Abs[z + step / 2]] - GetFromSolc[Abs[x], Abs[z - step / 2]]) /
    step;
  {Er, Ez}
 ]
```

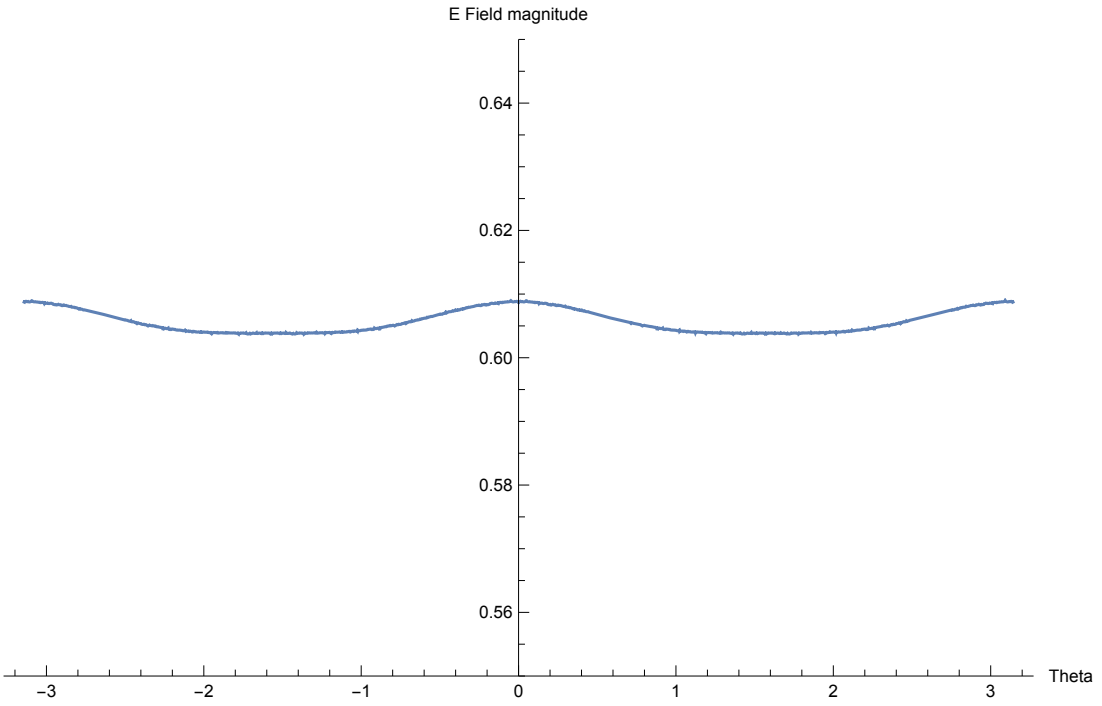
In[ ]:=
```
efieldvecs = Table[
  {{(a + gs) Cos[theta], (a + gs) Sin[theta]}, {(a + gs) Cos[theta], (a + gs) Sin[theta]} +
    {-GetEField[1, (a + gs) Cos[theta], 0, (a + gs) Sin[theta]][[1]],
     -GetEField[1, (a + gs) Cos[theta], 0, (a + gs) Sin[theta]][[3]]}},
  {theta, -Pi, Pi, Pi / 10}];
Show[Graphics[{Thickness[0.001], Arrowheads[0.02], Arrow[efieldvecs]}],
 Graphics[{Opacity[0.1], Disk[{0, 0}, a], Rectangle[{-b, -h / 2}, {b, h / 2}]}]]
```

Out[ ]=



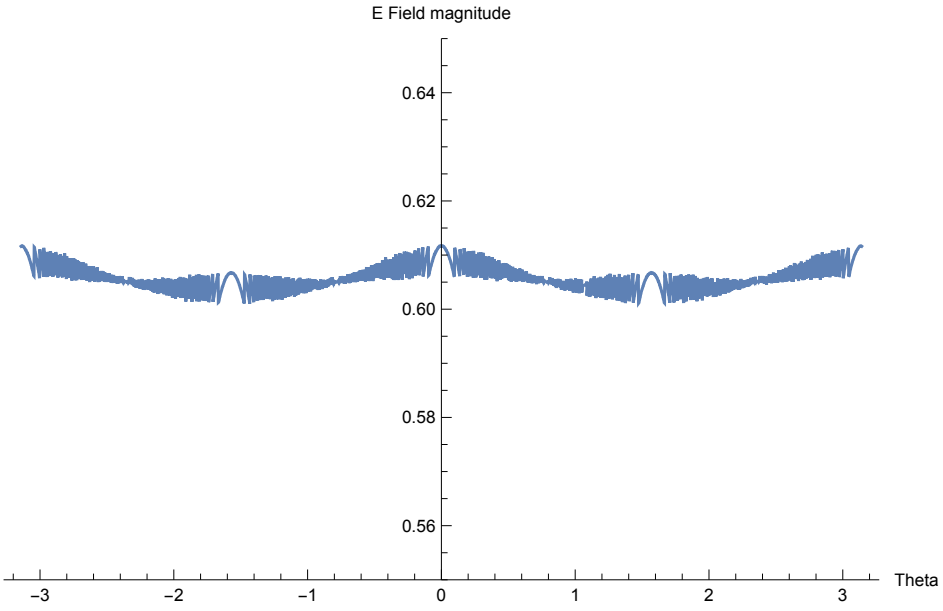
In[ ]:=
```
d = 0.1;
Plot[Norm[EstimateEField[(a + d) Cos[theta], (a + d) Sin[theta]]], {theta, -Pi, Pi},
 PlotRange → {0.55, 0.65}, AxesLabel → {"Theta", "E Field magnitude"}]
```

Out[ ]=



In[ ]:=
```
Plot[Norm[GetEField[1, (a + d) Cos[theta], 0, (a + d) Sin[theta]]], {theta, -Pi, Pi},
 PlotRange → {0.55, .65}, AxesLabel → {"Theta", "E Field magnitude"}]
```

Out[ ]=

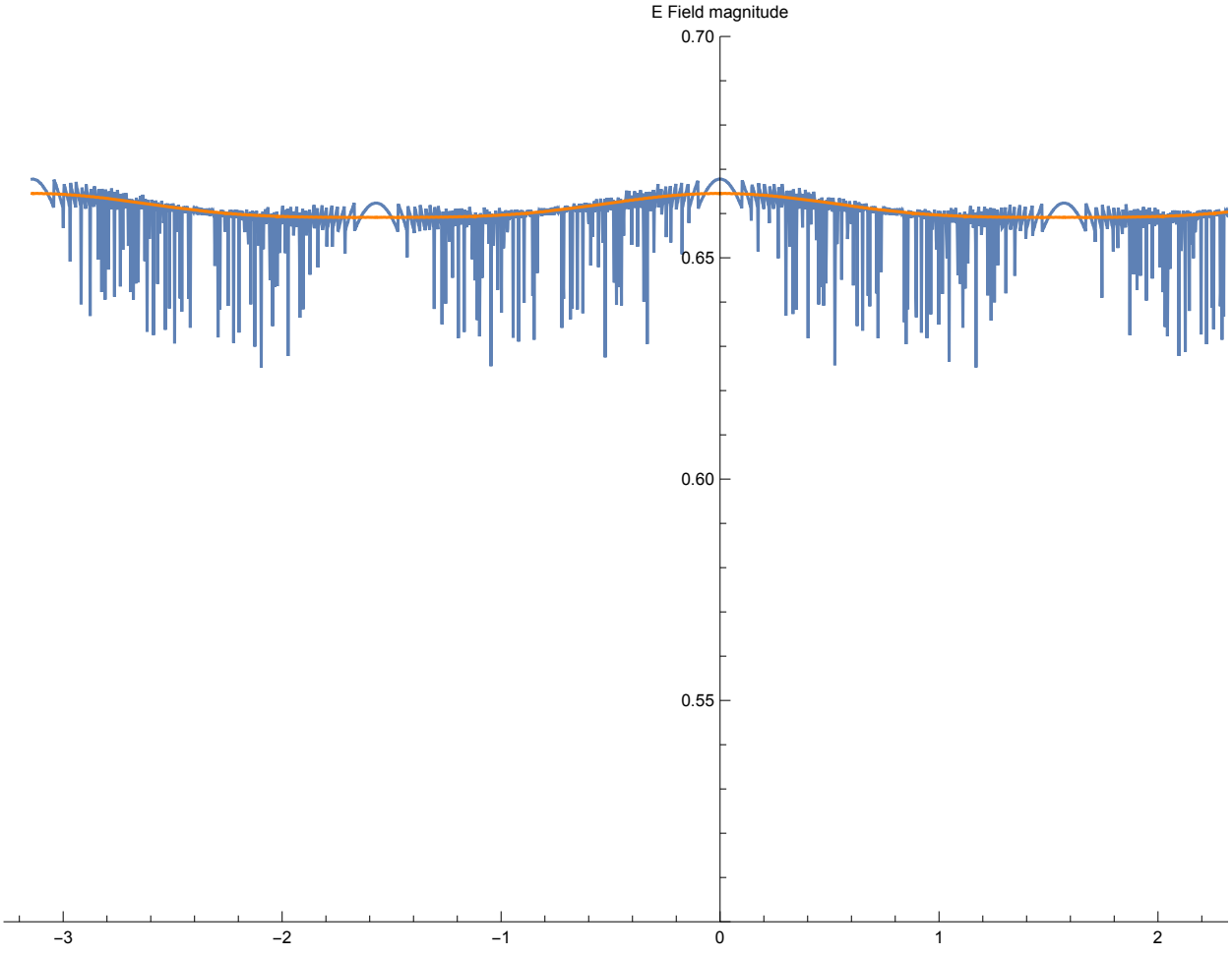


In[ ]:=
```
d = 1;
errors = Table[(Norm[EstimateEField[(a + d) Cos[theta], (a + d) Sin[theta]]] -
       Norm[GetEField[1, (a + d) Cos[theta], 0, (a + d) Sin[theta]]]) /
     Norm[EstimateEField[(a + d) Cos[theta], (a + d) Sin[theta]]],
    {theta, -Pi, Pi, Pi / 20000}];
100 * Max[errors]
```

Out[ ]=
```
0.32659
```

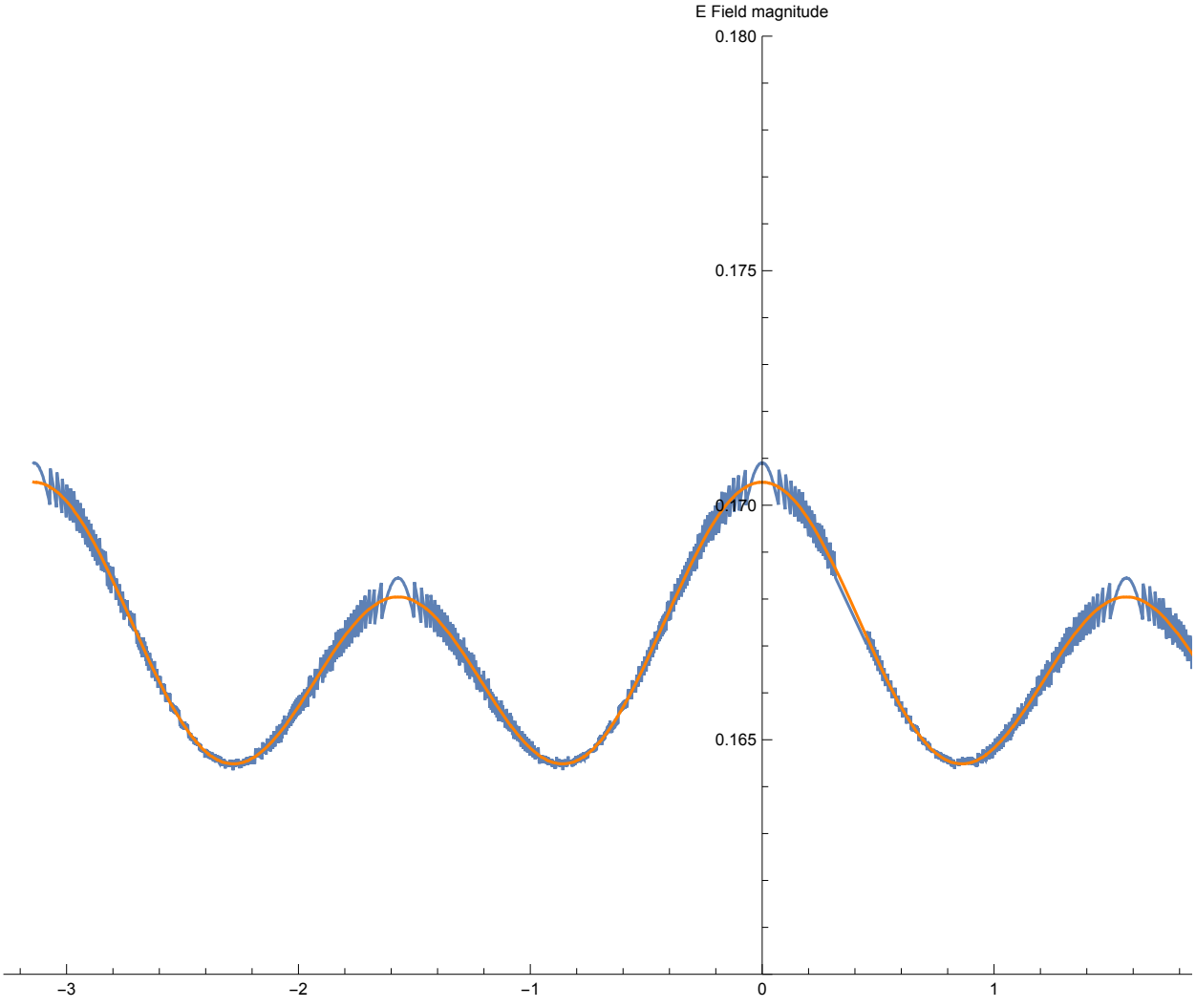
In[ ]:=
```
d = 0.01;
Show[Plot[Norm[GetEField[1, (a + d) Cos[theta], 0, (a + d) Sin[theta]]],
  {theta, -Pi, Pi}, PlotRange → {0.5, 0.7}],
 Plot[Norm[EstimateEField[(a + d) Cos[theta], (a + d) Sin[theta]]], {theta, -Pi, Pi},
  PlotStyle → Orange], AxesLabel → {"Theta", "E Field magnitude"}]
```

Out[ ]=



In[ ]:=
```
d = 2;
Show[Plot[Norm[GetEField[1, (a + d) Cos[theta], 0, (a + d) Sin[theta]]],
  {theta, -Pi, Pi}, PlotRange → {0.16, 0.18}],
 Plot[Norm[EstimateEField[(a + d) Cos[theta], (a + d) Sin[theta]]], {theta, -Pi, Pi},
  PlotStyle → Orange], AxesLabel → {"Theta", "E Field magnitude"}]
```

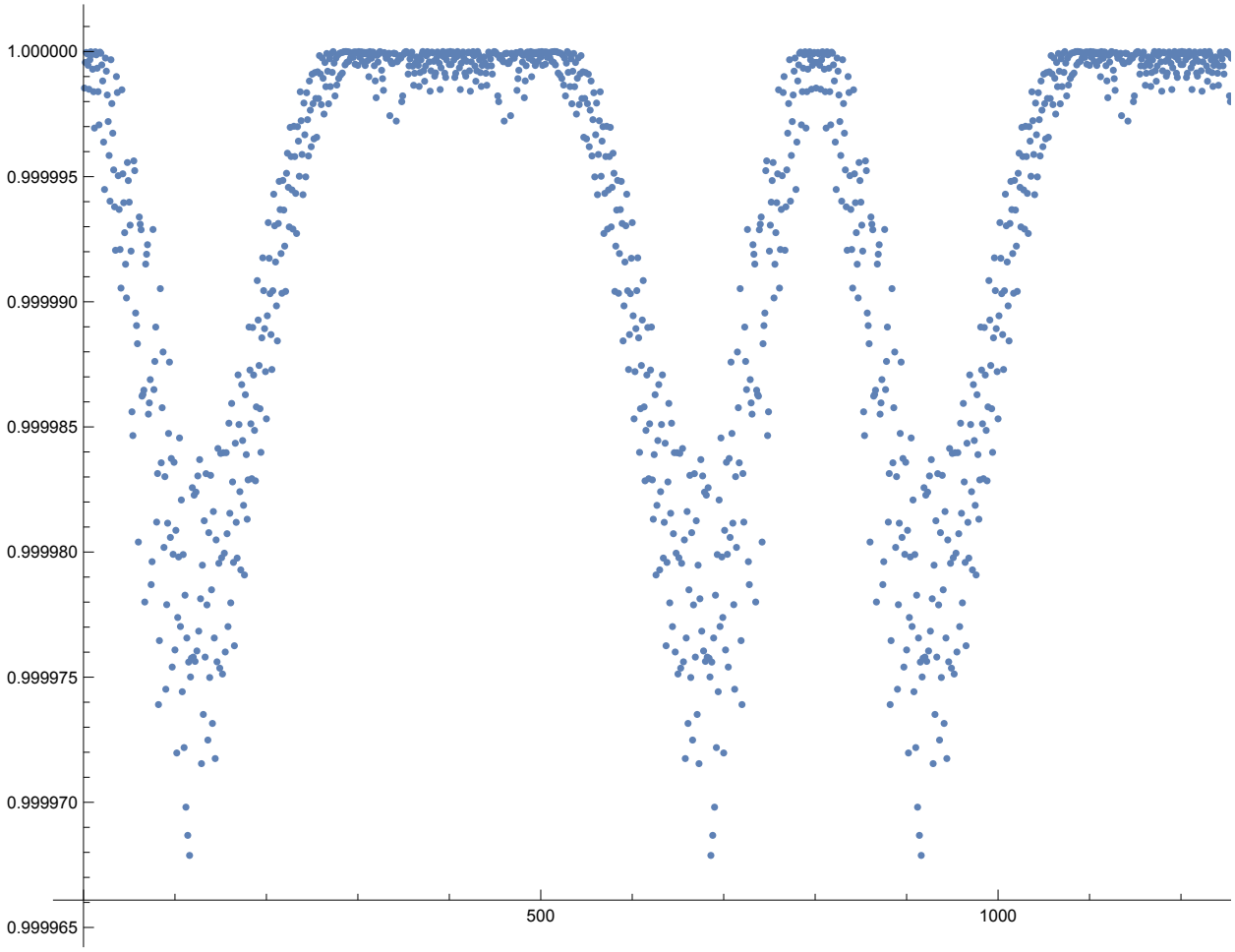
Out[ ]=



In[ ]:=
```
d = 1
directions = Table[(GetEField[1, (a + d) Cos[theta], 0, (a + d) Sin[theta]].
      {Cos[theta], 0, Sin[theta]}) /
    Norm[GetEField[1, (a + d) Cos[theta], 0, (a + d) Sin[theta]]], {theta,
    -Pi, Pi, 2 Pi / 1600}];
ListPlot[directions]
```

Out[ ]= 1

Out[ ]=

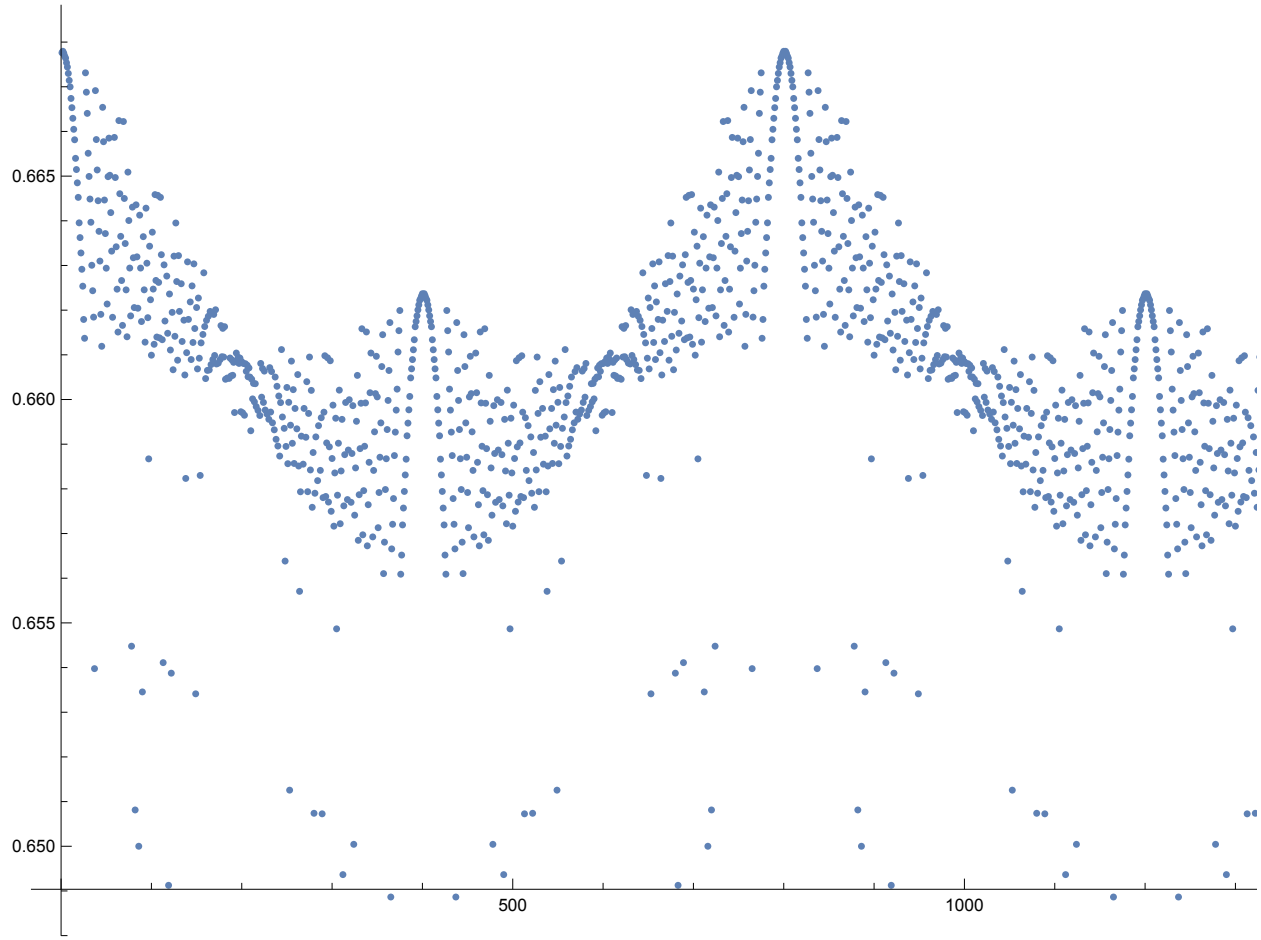


In[ ]:=
```
(*degrees of inaccuracy*)
ArcCos[Min[directions]] 180 / Pi
```

Out[ ]= 0.459241

In[ ]:=
```
mags = Table[Norm[GetEField[1, (a + gs) Cos[theta], 0, (a + gs) Sin[theta]]],
    {theta, -Pi, Pi, 2 Pi / 1600}];
ListPlot[mags]
```

Out[ ]=



In[ ]:=
```
testPoints = {{4.608842669337616`, 6.563585696985424`},
    {3.707180200517526`, 2.176750103748627`},
    {3.792037566239333`, 4.787781408276944`}, {3.855778982477084`,
     6.2237642461949205`}, {0.5816942256367047`, 6.431626072609152`},
    {1.866278784790517`, 1.5856972073106224`}, {2.905554076101517`,
     0.3042719406065806`}, {4.575278216079155`, 4.368535882641796`},
    {0.15228864787249807`, 6.7965613936293`}, {3.0835338797819465`,
     1.126015266510481`}, {0.8357538179378818`, 4.235467354793811`},
    {1.838948320701085`, 4.892962053921949`}, {0.6921786231648643`,
     1.984597266088046`}, {1.6282948913230884`, 6.1271049030900615`},
    {6.260241423248773`, 4.049899219509415`}, {4.314099980656643`,
     6.682551100251801`}, {3.13939323464767`, 3.2866671843639033`},
    {2.5535700266727304`, 6.458169956398989`}, {6.4600376108498825`,
     2.845434570939234`}, {5.915655343265746`, 5.078591103153024`}};
```

```
In[•]:= testResults = Table[
          Module[{e1, e2, n1, n2, diff, cos},
           e1 = -{GetEField[1, point[[1]], 0, point[[2]]][[1]],
              GetEField[1, point[[1]], 0, point[[2]]][[3]]};
           e2 = EstimateEField[point[[1]], point[[2]]];
           n1 = Norm[e1];
           n2 = Norm[e2];
           diff = 2 (n1 - n2) / (n1 + n2);
           cos = e1.e2 / (n1 n2);
           {point, e1, e2, n1, n2, diff, cos }
          ], {point, testPoints}];
        testResults = Insert[testResults, {"Point", "E from File",
           "E from Laplace", "Norm 1", "Norm 2", "Difference", "Cos"}, 1];
        Print[phiFile];
        Grid[testResults, Frame → All, Spacings → 2]
```

Phi file: 2/5/2023 Phi from 2D cylindrical solve

Point	E from File	E from Laplace	Norm 1	Norm 2	Difference	Cos
{4.60884, 6.56359}	{-0.00958667, -0.0314003}	{-0.00958494, -0.0314021}	0.0328311	0.0328324	-0.0000379209	1.
{3.70718, 2.17675}	{-0.125929, -0.0692497}	{-0.125842, -0.0692609}	0.143714	0.143643	0.000492979	1.
{3.79204, 4.78778}	{-0.0390244, -0.0523992}	{-0.0390244, -0.052376}	0.0653344	0.0653158	0.00028492	1.
{3.85578, 6.22376}	{-0.0145572, -0.0426574}	{-0.0145568, -0.0426622}	0.0450728	0.0450773	-0.0000997828	1.
{0.581694, 6.43163}	{-0.0031235, -0.0775592}	{-0.0031 0674, -0.0775946}	0.0776221	0.0776567	-0.000446382	1.
{1.86628, 1.5857}	{-0.339956, -0.287458}	{-0.339769, -0.287402}	0.445199	0.44502	0.000403887	1.

Out[ ]=

{2.90555, 0.304272}	{-0.313846, -0.0323101}	{-0.313743, -0.0322276}	0.315505	0.315394	0.000350693	1.
{4.57528, 4.36854}	{-0.0442936, -0.0390171}	{-0.0442909, -0.0390052}	0.0590275	0.0590177	0.000166391	1.
{0.152289, 6.79656}	{-0.000472751, -0.0755912}	{-0.000464495, -0.0755819}	0.0755926	0.0755833	0.000123898	1.
{3.08353, 1.12602}	{-0.234641, -0.083352}	{-0.234826, -0.083882}	0.249006	0.249192	-0.000749257	1.
{0.835754, 4.23547}	{-0.0265261, -0.142051}	{-0.0265446, -0.142025}	0.144506	0.144484	0.000153714	1.
{1.83895, 4.89296}	{-0.0306978, -0.0936121}	{-0.0307348, -0.0936649}	0.0985169	0.0985786	-0.000625871	1.
{0.692179, 1.9846}	{-0.19912, -0.569007}	{-0.198592, -0.569215}	0.602842	0.602863	-0.000036235	1.
{1.62829, 6.1271}	{-0.010915, -0.0733952}	{-0.010932, -0.0733701}	0.0742024	0.07418	0.000301489	1.
{6.26024, 4.0499}	{-0.0388513, -0.0123728}	{-0.0388805, -0.0123718}	0.0407739	0.0408013	-0.000672549	1.
{4.3141, 6.68255}	{-0.00811807, -0.0351415}	{-0.00811806, -0.0351454}	0.036067	0.0360708	-0.000107719	1.

{3.13939, 3.28667}	{-0.0881766, -0.0900567}	{-0.0881195, -0.0900236}	0.126037	0.125973	0.000505233	1.
{2.55357, 6.45817}	{-0.0101184, -0.0595603}	{-0.0101166, -0.0595414}	0.0604137	0.0603947	0.000314094	1.
{6.46004, 2.84543}	{-0.0527552, -0.0092606054}	{-0.0528024, -0.0092606173}	0.0535619	0.0536085	-0.000870031	1.
{5.91566, 5.07859}	{-0.0269175, -0.0162529}	{-0.0269149, -0.016249}	0.0314437	0.0314395	0.000132705	1.

In[•]:=
```
testResultsv2 = Table[{
      point,
      -GetPotential[1, point[[1]], 0, point[[2]]],
      solc[point[[1]], point[[2]]],
      100 * (solc[point[[1]], point[[2]]] + GetPotential[1, point[[1]], 0, point[[2]]]) /
        solc[point[[1]], point[[2]]]} // N,
   {point, testPoints}];
testResultsv2 = Insert[testResultsv2, {"Point",
    "Potential from File", "Potential from Laplace", "Percent Error"}, 1];
Grid[testResultsv2, Frame → All, Spacings → 2]
```

Out[ ]=

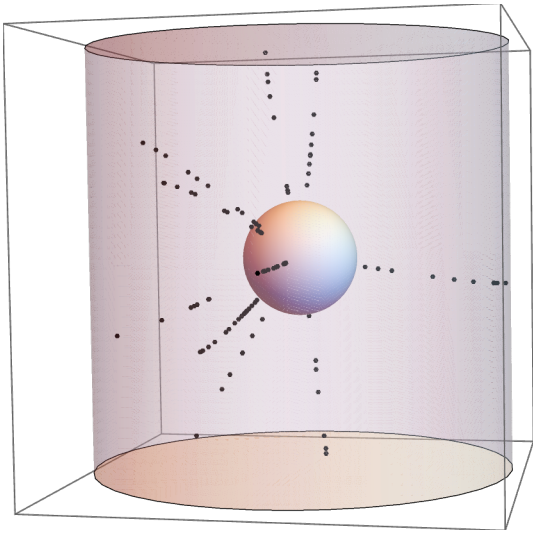
Point	Potential from File	Potential from Laplace	Percent Error
{4.60884, 6.56359}	−0.0228437	−0.0228436	−0.0000893453
{3.70718, 2.17675}	−0.285638	−0.285638	−0.000119639
{3.79204, 4.78778}	−0.113569	−0.113568	−0.0000966593
{3.85578, 6.22376}	−0.0444733	−0.0444733	−0.0000920436
{0.581694, 6.43163}	−0.0647829	−0.0647828	−0.0000635923
{1.86628, 1.5857}	−0.755247	−0.755245	−0.000283062
{2.90555, 0.304272}	−0.576356	−0.576355	−0.000247503
{4.57528, 4.36854}	−0.0999098	−0.0999097	−0.000092579
{0.152289, 6.79656}	−0.0375394	−0.0375394	−0.000113133
{3.08353, 1.12602}	−0.476133	−0.476132	−0.000172861
{0.835754, 4.23547}	−0.28458	−0.284579	−0.000140282
{1.83895, 4.89296}	−0.177993	−0.177992	−0.000109267
{0.692179, 1.9846}	−0.935483	−0.935479	−0.000463012
{1.62829, 6.1271}	−0.0807665	−0.0807664	−0.000110662
{6.26024, 4.0499}	−0.0374481	−0.0374481	-5.93771×10^{-6}
{4.3141, 6.68255}	−0.0214964	−0.0214964	−0.0000969176
{3.13939, 3.28667}	−0.256277	−0.256277	−0.0000666832
{2.55357, 6.45817}	−0.0487486	−0.0487486	−0.0000427234
{6.46004, 2.84543}	−0.0414332	−0.0414332	0.0000257004
{5.91566, 5.07859}	−0.0340119	−0.0340119	−0.000136803

In[ ]:=
```
points = Reap[AvgElectronsEulerSC[53 / 100, 1, 15, 2 / 5, 99 / 100, True, 10]][[2]]
```

```
1000 * RMS travel error: 1.77026

V/NC 1.5

λ: 53/100

Geometry: SC

Collisions: 10.0 ± 1.3

Mean path: 8.8186
```

In[ ]:=
```
Show[Graphics3D[ {Opacity[0.3], Cylinder[{{0, 0, -7.3}, {0, 0, 7.3}}, 7.3]}],
 Graphics3D[Ball[{0, 0, 0}, 2]],
 Graphics3D[{Black, PointSize[0.01], Point[points[[1]]]}]]
```

Out[ ]=



Sphere-in-cylinder MC's

In[ ]:=
```
AvgElectronsEulerSC[λ_, Ni_, Ui_, dt_, tbound_, print_, getbins_, count_] :=
 Module[{eErrorList, V, result, stack, collisions, paths, svals, countbin,
   electronsbin, squaredbin, lbin, lbinSquared, data, seedcount, totalpaths, Nc},
  V = Ui * Ni;
  totalpaths = 0;
  seedcount = 1;
  collisions = {};
  eErrorList = {};
  paths = {};
  svals = 0;
  If[getbins,
   countbin = Table[0, 2 / binwidth];
   electronsbin = Table[0, 2 / binwidth];
   squaredbin = Table[0, 2 / binwidth];
   lbin = Table[0, lmax / 2 + 1];
   lbinSquared = Table[0, lmax / 2 + 1, lmax / 2 + 1]];
  data = {};
  stack = CreateDataStructure["Stack"];
  result = Table[
    Module[{x0, x1, y0, y1, z0, z1, numElectrons, cosθ, ϕ,
      s, vx, vy, vz, energy, eError, numCollisions, pathval, sval, idx,
      cθ0, ϕ0, SH, l, l2, collisionE, cost, phi, start, path, input},
     numElectrons = 1;
```

```
numCollisions = 0;
stack["Push", GetStartingPos[]];
{cθ0, ϕ0} = GetAngles[x0, y0, z0];
While[! stack["EmptyQ"],
 {x0, y0, z0, vx, vy, vz} = stack["Pop"];
 cosθ = 1;
 ϕ = 0;
 energy = 0;
 path = 0;
 While[
  s = RandomVariate[ExponentialDistribution[1 / λ], WorkingPrecision → 16];
  {x1, y1, z1, vx, vy, vz, energy, eError, pathval, sval} =
   GetNewPositionsEuler[dt, s, V, x0, y0, z0, vx, vy, vz];
  AppendTo[eErrorList, eError];
  InBounds[x1, y1, z1],
  path += pathval;
  AppendTo[paths, pathval];
  svals += s;
  numCollisions++;
  If[energy ≥ Ui,
   numElectrons++;
   stack["Push", {x1, y1, z1, 0, 0, 0}];
   energy -= Ui,
   collisionE =
    SetPrecision[Ui * RandomVariate[UniformDistribution[{1 / 10, 1 / 2}]], 16];
   energy -= collisionE;
   If[energy < 0, energy = 0];
  ];
  {vx, vy, vz, cost, phi} = UpdateVelocity[energy, vx, vy, vz, tbound];
  {x0, y0, z0} = {x1, y1, z1};
 ];
 totalpaths += path;
];
AppendTo[data, {cθ0, numElectrons}];
If[getbins,
 idx = FindBinNum[cθ0, binwidth];
 countbin[[idx]] += 1;
 electronsbin[[idx]] += numElectrons;
 squaredbin[[idx]] += numElectrons^2;
 SH = numElectrons Table[
    Conjugate[SphericalHarmonicY[l, 0, ArcCos[cθ0], ϕ0]], {l, 0, lmax, 2}];
 For[l = 0, l ≤ lmax, l += 2,
  lbin[[l / 2 + 1]] += SH[[l / 2 + 1]];
  For[l2 = 0, l2 ≤ lmax, l2 += 2,
```

```
            lbinSquared[[l / 2 + 1, l2 / 2 + 1]] += SH[[l / 2 + 1]] × SH[[l2 / 2 + 1]]
           ]
          ];
         ];
         AppendTo[collisions, numCollisions];
         numElectrons],
        {count}];
      If[print,
       Print["1000 * RMS travel error: ", 1000 * RootMeanSquare[eErrorList]];
       Print["V/NC ", Ni Ui λ / 5.3];
       Print["λ: ", λ];
       Print["Geometry: ", ID];
       Print["Collisions: ",
        Around[Mean[collisions], StandardDeviation[collisions] / Sqrt[count]]];
       Print["Mean path: ",  totalpaths / count / λ];
       Print[phiFile]
      ];
      Sow[data ];
      Around[Mean[result], StandardDeviation[result] / Sqrt[count]]]
```

Creating a faster version to run in-parallel

In[ ]:=
```
AbsoluteTiming[AvgElectronsEulerSCFAST[1 / 2, 10, 15, 2 / 5, 4 / 5, True, False, 100]]
```

```
1000 * RMS travel error: 14.6545

V/NC 14.1509

λ: 1/2

Geometry: SC

Collisions: 226. ±10.

Mean path: 0

Phi file: 2/5/2023 Phi from 2D cylindrical solve
```

Out[ ]=
```
{115.108, 36.9 ±1.4}
```

In[ ]:=
```
AvgElectronsEulerSCFAST[λ_, Ni_, Ui_, dt_, tbound_, print_, getbins_, count_] :=
 Module[{masterdata, eErrorList, V, result, stack, collisions,
   paths, svals, eErrorListMASTER, countbin, electronsbin,
   squaredbin, lbin, lbinSquared, seedcount, totalpaths, Nc},
  V = Ui * Ni;
  seedcount = 1;
  collisions = {};
  (*each kernel creates its own version of eErrorList*)
  eErrorListMASTER = {};
  paths = {};
```

```
svals = 0;
If[getbins,
 countbin = Table[0, 2 / binwidth];
 electronsbin =  Table[0, 2 / binwidth];
 squaredbin = Table[0, 2 / binwidth];
 lbin = Table[0, lmax / 2 + 1];
 lbinSquared =  Table[0, lmax / 2 + 1, lmax / 2 + 1]];
masterdata = {};
result = ParallelTable[
  Module[{x0, x1, y0, y1, z0, z1, numElectrons, cosθ, ϕ, s, vx,
    vy, vz, energy, data, eError, numCollisions, pathval, sval, idx,
    cθ0, ϕ0, SH, l, l2, collisionE, cost, phi, start, path, input},
   numElectrons = 1;
   numCollisions = 0;
   eErrorList = {};
   stack = CreateDataStructure["Stack"];
   stack["Push", GetStartingPos[]];
   {cθ0, ϕ0} = GetAngles[x0, y0, z0];
   While[! stack["EmptyQ"],
    {x0, y0, z0, vx, vy, vz} = stack["Pop"];
    cosθ = 1;
    ϕ = 0;
    energy = 0;
    data = {};
    path = 0;
    While[
     s = RandomVariate[ExponentialDistribution[1 / λ], WorkingPrecision → 16];
     {x1, y1, z1, vx, vy, vz, energy, eError, pathval, sval} =
      GetNewPositionsEuler[dt, s, V, x0, y0, z0, vx, vy, vz];
     AppendTo[eErrorList, eError];
     InBounds[x1, y1, z1],
     path += pathval;
     AppendTo[paths, pathval];
     svals += s;
     numCollisions++;
     If[energy ≥ Ui,
      numElectrons++;
      stack["Push", {x1, y1, z1, 0, 0, 0}];
      energy -= Ui,
      collisionE =
       SetPrecision[Ui * RandomVariate[UniformDistribution[{1 / 10, 1 / 2}]], 16];
      energy -= collisionE;
      If[energy < 0, energy = 0];
     ];
```

```
      {vx, vy, vz, cost, phi} = UpdateVelocity[energy, vx, vy, vz, tbound];
      {x0, y0, z0} = {x1, y1, z1};
     ];
    ];
    AppendTo[data, {cθ0, numElectrons}];
    If[getbins,
     idx = FindBinNum[cθ0, binwidth];
     countbin[[idx]] += 1;
     electronsbin[[idx]] += numElectrons;
     squaredbin[[idx]] += numElectrons^2;
     SH = numElectrons Table[
        Conjugate[SphericalHarmonicY[l, 0, ArcCos[cθ0], ϕ0]], {l, 0, lmax, 2}];
     For[l = 0, l ≤ lmax, l += 2,
      lbin[[l / 2 + 1]] += SH[[l / 2 + 1]];
      For[l2 = 0, l2 ≤ lmax, l2 += 2,
       lbinSquared[[l / 2 + 1, l2 / 2 + 1]] += SH[[l / 2 + 1]] × SH[[l2 / 2 + 1]]
      ]
     ];
    ];
    {numElectrons, eErrorList, numCollisions, Last[data]}],
   {count}];
collisions = Flatten[#[[3]] & /@ result];
eErrorListMASTER = Flatten[#[[2]] & /@ result];
masterdata = #[[4]] & /@ result;
result = Flatten[#[[1]] & /@ result];
If[print,
 Print["1000 * RMS travel error: ",
  1000 * RootMeanSquare[Flatten[eErrorListMASTER]]];
 Print["V/NC ", Ni Ui λ / 5.3];
 Print["λ: ", λ];
 Print["Geometry: ", ID];
 Print["Collisions: ",
  Around[Mean[collisions], StandardDeviation[collisions] / Sqrt[count]]];
 Print[phiFile]
];
Sow[masterdata];
Around[Mean[result], StandardDeviation[result] / Sqrt[count]]]
```

Playing with spherical harmonics (using standardized testing data)

In[•]:=
```
Import["/Users/wbowers/Downloads/testData1.txt", "Table"]
```

Out[ ]=

```
{{Command, line:, emc, Ni=11.3, Nc=10, Ui=15, reps=10000, dt=1e-3, scatter=0.8,
   seed=1, shape=scf, phifile=cwPhi.txt, showpath=true, pathnum=-2},
  {}, {-0.24765, 121}, {0.58485, 30}, {0.82089, 64}, {0.51459, 60},
  {0.53806, 47}, {-0.75322, 74}, {-0.39226, 42}, {-0.80142, 69}, {0.51232, 13},
  {-0.17224, 62}, {-0.15049, 37}, {-0.57335, 67}, {0.66387, 29},
  {0.95469, 35}, {0.29807, 100}, ··· 9968 ···, {0.25002, 37}, {-0.98531, 44},
  {-0.48698, 34}, {0.99284, 27}, {0.27253, 84}, {-0.37638, 49}, {0.43035, 20},
  {-0.37275, 37}, {-0.36154, 33}, {-0.33106, 32}, {0.40546, 16}, {0.35234, 68},
  {-0.67814, 12}, {0.12916, 37}, {-0.18555, 14}, {-0.93947, 85}, {0.26674, 68}}
```

large output | show less | show more | show all | set si | ze limit...

In[ ]:=

```
SHtestdata = Drop[Import["/Users/wbowers/Downloads/testData1.txt", "Table"], 2];
SHtestdata2 = Drop[Import["/Users/wbowers/Downloads/testData2.txt", "Table"], 2];
```

In[ ]:=

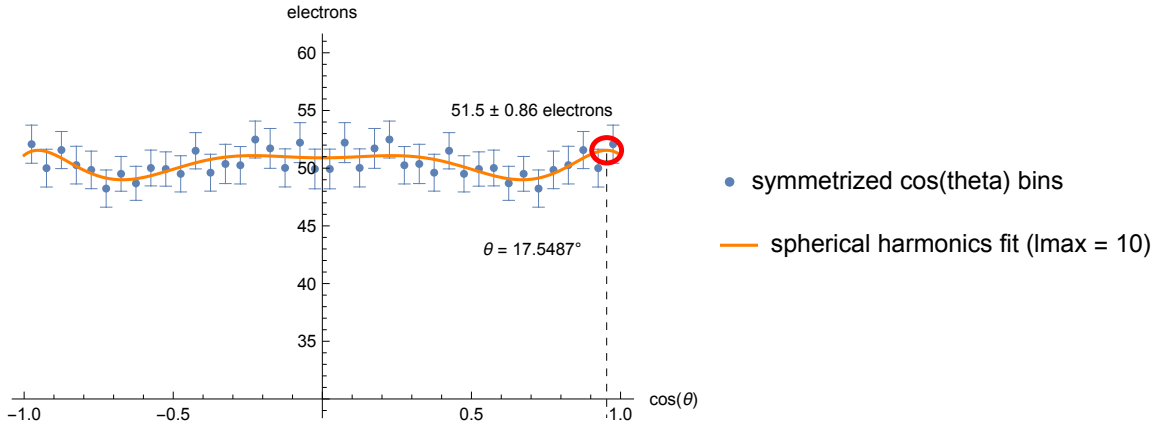
```
MakeElectronsPlot[data_] :=
 Module[{tenthOrderFit, SHYs, max, error, res, electrons, electronsSquared,
   counts, finalbins, half, temp, symbins, symcounts,
   symsquaredbins, errors, finalsymbins, cΘ0, numelectrons},
  tenthOrderFit = LinearModelFit[data,
    Table[SphericalHarmonicY[l, 0, t, 0], {l, 0, 10, 2}] /. Cos[t] → z,
    z, IncludeConstantBasis → False];
  SHYs = Table[SphericalHarmonicY[l, 0, t, 0], {l, 0, lmax, 2}] /. {Cos[t] → z};
  max = NMaximize[{tenthOrderFit[z], -1 ≤ z ≤ 1}, z];
  error =
   Sqrt[SHYs.tenthOrderFit["CovarianceMatrix"].Transpose[SHYs]] /. max[[2]];
  res = Around[max[[1]], error];
  electrons = Table[0, 2 / binwidth];
  electronsSquared = Table[0, 2 / binwidth];
  counts = Table[0, 2 / binwidth];
  For[i = 1, i ≤ Length[data], i++,
   {cΘ0, numelectrons} = data[[i]];
   idx = FindBinNum[cΘ0, binwidth];
   electrons[[idx]] += numelectrons;
   electronsSquared[[idx]] += numelectrons^2;
   counts[[idx]] += 1
  ];
  finalbins = electrons / counts;
  finalbins =
   Transpose[{Table[theta, {theta, -0.975, 0.975, binwidth}], finalbins}];

  half = Length[finalbins] / 2;
  temp = #[[2]] & /@ finalbins;
```

```
symbins = (temp[[1 ;; half]] + Reverse[temp[[half + 1 ;; Length[temp]]]]) / 2;
symcounts = (counts[[1 ;; half]] + Reverse[counts[[half + 1 ;; Length[counts]]]]) / 2;
symsquaredbins = (electronsSquared[[1 ;; half]] +
    Reverse[electronsSquared[[half + 1 ;; Length[electronsSquared]]]]) / 2;

symbins = Flatten[AppendTo[symbins, Reverse[symbins]]];
symcounts = Flatten[AppendTo[symcounts, Reverse[symcounts]]];
symsquaredbins = Flatten[AppendTo[symsquaredbins, Reverse[symsquaredbins]]];

symbins = Transpose[{Table[theta, {theta, -0.975, 0.975, binwidth}], symbins}];
symsquaredbins =
 Transpose[{Table[theta, {theta, -0.975, 0.975, binwidth}], symsquaredbins}];

errors = Sqrt[(#[[2]] & /@ symsquaredbins) / symcounts - (#[[2]] & /@ symbins)^2] /
  Sqrt[symcounts];
finalsymbins = symbins;

Print[finalbins];

For[i = 1, i ≤ Length[errors], i++,
 finalsymbins[[i]] = {symbins[[i, 1]], Around[symbins[[i, 2]], errors[[i]]]}];
Show[ListPlot[finalsymbins, PlotLegends → {"symmetrized cos(theta) bins"}],
 Plot[tenthOrderFit[x], {x, -1, 1}, PlotStyle → Orange,
  PlotLegends → {"spherical harmonics fit (lmax = 10)"}],
 Graphics[{Red, Thickness[0.008], Circle[{z /. max[[2]], max[[1]]}, {0.05, 1.1}]}],
 Graphics[{Text[ToString[Round[res[[1]], 0.1]] <> " ± " <>
     ToString[Round[res[[2]], 0.01]] <> " electrons", {(z /. max[[2]]) - 0.25, 55}],
   Text["θ = " <> ToString[ArcCos[(z /. max[[2]])] * 1 / Degree] <> "°",
    {(z /. max[[2]]) - 0.25, 43}]}],
 Graphics[{Dashed, Black, Line[{{z /. max[[2]], 20}, {z /. max[[2]], max[[1]]}}]}],
 AxesOrigin → {0, 30}, PlotRange → {30, 60}, AxesLabel → {"cos(θ)", "electrons"}]
]
```

In[•]:=
```
fig6 = MakeElectronsPlot[SHtestdata2]
```

$$\left\{\left\{-0.975, \frac{13\,213}{251}\right\}, \left\{-0.925, \frac{12\,707}{249}\right\}, \left\{-0.875, \frac{13\,771}{270}\right\}, \left\{-0.825, \frac{13\,102}{257}\right\}, \left\{-0.775, \frac{5949}{124}\right\},\right.$$
$$\left\{-0.725, \frac{11\,422}{243}\right\}, \left\{-0.675, \frac{11\,583}{226}\right\}, \left\{-0.625, \frac{6272}{137}\right\}, \left\{-0.575, \frac{13\,790}{269}\right\}, \left\{-0.525, \frac{7031}{143}\right\},$$
$$\left\{-0.475, \frac{13\,244}{251}\right\}, \left\{-0.425, \frac{667}{13}\right\}, \left\{-0.375, \frac{12\,921}{253}\right\}, \left\{-0.325, \frac{12\,401}{240}\right\}, \left\{-0.275, \frac{1807}{35}\right\},$$
$$\left\{-0.225, \frac{12\,193}{244}\right\}, \left\{-0.175, \frac{417}{8}\right\}, \left\{-0.125, \frac{4049}{79}\right\}, \left\{-0.075, \frac{12\,774}{241}\right\}, \left\{-0.025, \frac{12\,330}{253}\right\},$$
$$\left\{0.025, \frac{6851}{134}\right\}, \left\{0.075, \frac{11\,982}{233}\right\}, \left\{0.125, \frac{6197}{127}\right\}, \left\{0.175, \frac{12\,313}{240}\right\}, \left\{0.225, \frac{15\,179}{276}\right\},$$
$$\left\{0.275, \frac{3275}{67}\right\}, \left\{0.325, \frac{11\,923}{243}\right\}, \left\{0.375, \frac{11\,893}{247}\right\}, \left\{0.425, \frac{6464}{125}\right\}, \left\{0.475, \frac{5273}{114}\right\},$$
$$\left\{0.525, \frac{4207}{83}\right\}, \left\{0.575, \frac{11\,851}{243}\right\}, \left\{0.625, \frac{13\,570}{263}\right\}, \left\{0.675, \frac{12\,372}{259}\right\}, \left\{0.725, \frac{11\,673}{236}\right\},$$
$$\left.\left\{0.775, \frac{12\,621}{244}\right\}, \left\{0.825, \frac{11\,744}{237}\right\}, \left\{0.875, \frac{12\,879}{247}\right\}, \left\{0.925, \frac{10\,726}{219}\right\}, \left\{0.975, \frac{1391}{27}\right\}\right\}$$

Out[ ]=



In[ ]:=
```
Export["/Users/wbowers/Documents/stem-fellowship/tif-figs/sh-fitting.TIFF",
 fig6, ImageResolution → 600]
```

Out[ ]= /Users/wbowers/Documents/stem-fellowship/tif-figs/sh-fitting.TIFF

In[ ]:=
```
r = Reap[AvgElectronsEulerSCFAST[53 / 34, 40, 15, 2 / 5, 4 / 5, True, False, 50 000]]
```

1000 * RMS travel error: 149.96

V/NC 176.471

λ: $\frac{53}{34}$

Geometry: SC

Collisions: 42.47 ± 0.20

Phi file: 2/5/2023 Phi from 2D cylindrical solve

Out[•]=

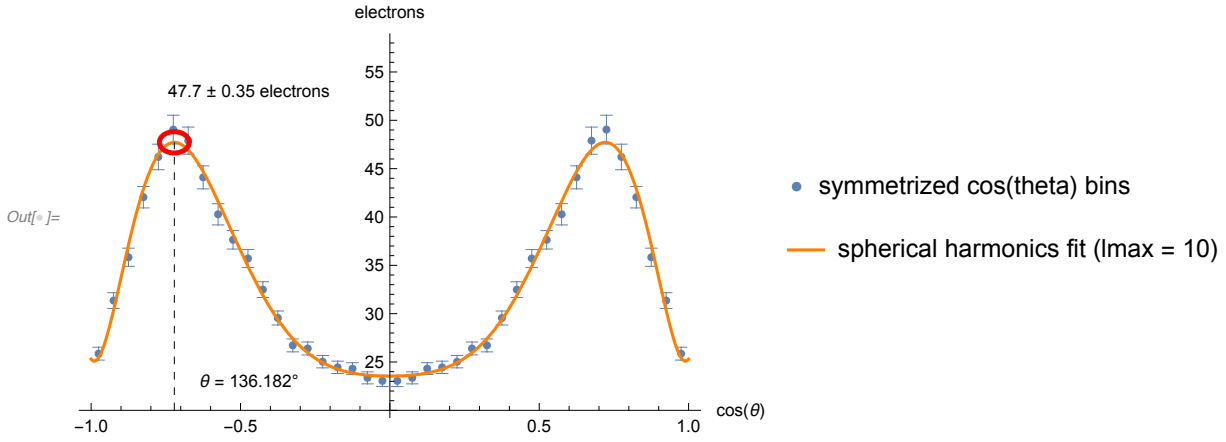
```
{32.92 ± 0.15, {{{{0.360795, 71}, {0.598697, 88}, {-0.974832, 38}, {-0.767807, 18},
     {-0.489686, 17}, {0.7662, 20}, {0.151377, 6}, {0.417244, 12},
     {-0.205477, 45}, {-0.0468744, 14}, {-0.109721, 19}, {-0.530765, 50},
     ··· 49 977 ···, {0.275548, 17}, {0.411143, 40}, {0.138401, 5},
     {0.791972, 11}, {-0.960341, 7}, {0.774839, 4}, {0.908373, 22},
     {0.861046, 62}, {-0.448823, 5}, {-0.865586, 27}, {-0.119944, 21}}}}}
```

large output | show less | show more | show all | set si | ze limit...

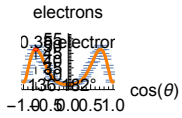
In[ ]:= **MakeElectronsPlot[r[[2, 1, 1]]]**

$$\left\{\left\{-0.975, \frac{103}{4}\right\}, \left\{-0.925, \frac{42\,773}{1344}\right\}, \left\{-0.875, \frac{15\,397}{421}\right\}, \left\{-0.825, \frac{51\,929}{1243}\right\}, \left\{-0.775, \frac{3556}{75}\right\},\right.$$
$$\left\{-0.725, \frac{49\,355}{1018}\right\}, \left\{-0.675, \frac{8771}{178}\right\}, \left\{-0.625, \frac{25\,888}{583}\right\}, \left\{-0.575, \frac{49\,362}{1237}\right\}, \left\{-0.525, \frac{25\,157}{652}\right\},$$
$$\left\{-0.475, \frac{44\,791}{1250}\right\}, \left\{-0.425, \frac{43\,137}{1366}\right\}, \left\{-0.375, \frac{41\,359}{1373}\right\}, \left\{-0.325, \frac{36\,101}{1330}\right\}, \left\{-0.275, \frac{17\,747}{670}\right\},$$
$$\left\{-0.225, \frac{10\,755}{443}\right\}, \left\{-0.175, \frac{16\,970}{683}\right\}, \left\{-0.125, \frac{4624}{193}\right\}, \left\{-0.075, \frac{30\,052}{1285}\right\}, \left\{-0.025, \frac{4984}{213}\right\},$$
$$\left\{0.025, \frac{29\,181}{1286}\right\}, \left\{0.075, \frac{28\,301}{1213}\right\}, \left\{0.125, \frac{16\,602}{673}\right\}, \left\{0.175, \frac{32\,179}{1338}\right\}, \left\{0.225, \frac{34\,483}{1337}\right\},$$
$$\left\{0.275, \frac{11\,339}{431}\right\}, \left\{0.325, \frac{34\,946}{1329}\right\}, \left\{0.375, \frac{39\,847}{1375}\right\}, \left\{0.425, \frac{10\,718}{321}\right\}, \left\{0.475, \frac{44\,224}{1243}\right\},$$
$$\left\{0.525, \frac{46\,561}{1269}\right\}, \left\{0.575, \frac{16\,627}{409}\right\}, \left\{0.625, \frac{17\,034}{389}\right\}, \left\{0.675, \frac{45\,491}{978}\right\}, \left\{0.725, \frac{3323}{67}\right\},$$
$$\left.\left\{0.775, \frac{15\,707}{349}\right\}, \left\{0.825, \frac{24\,671}{583}\right\}, \left\{0.875, \frac{7651}{218}\right\}, \left\{0.925, \frac{20\,167}{653}\right\}, \left\{0.975, \frac{11\,404}{439}\right\}\right\}$$

Out[ ]=



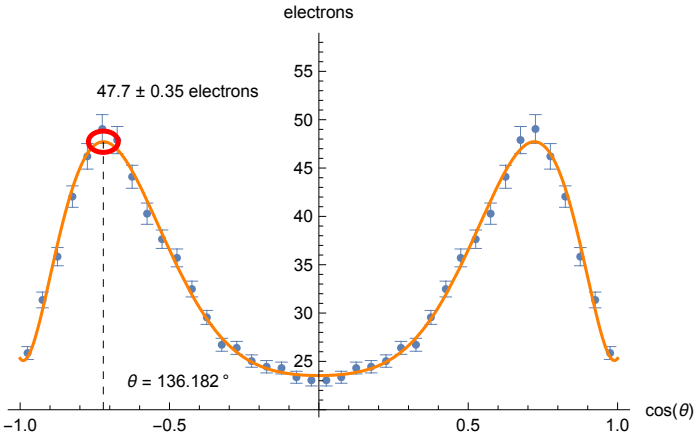
In[ ]:= p1 =



symmetrized cos(theta) bins

spherical harmonics fit (lmax = 10)

Out[ ]=



symmetrized cos(theta) bins

spherical harmonics fit (lmax = 10)

In[ ]:= r2 = Reap[AvgElectronsEulerSCFAST[53 / 700, 171 / 10, 15, 2 / 5, 4 / 5, True, False, 5000]]

1000 * RMS travel error: 21.1145

V/NC 3.66429

λ: $\frac{53}{700}$

Geometry: SC

Collisions: 2832. ± 14.

Phi file: 2/5/2023 Phi from 2D cylindrical solve

Out[ ]=

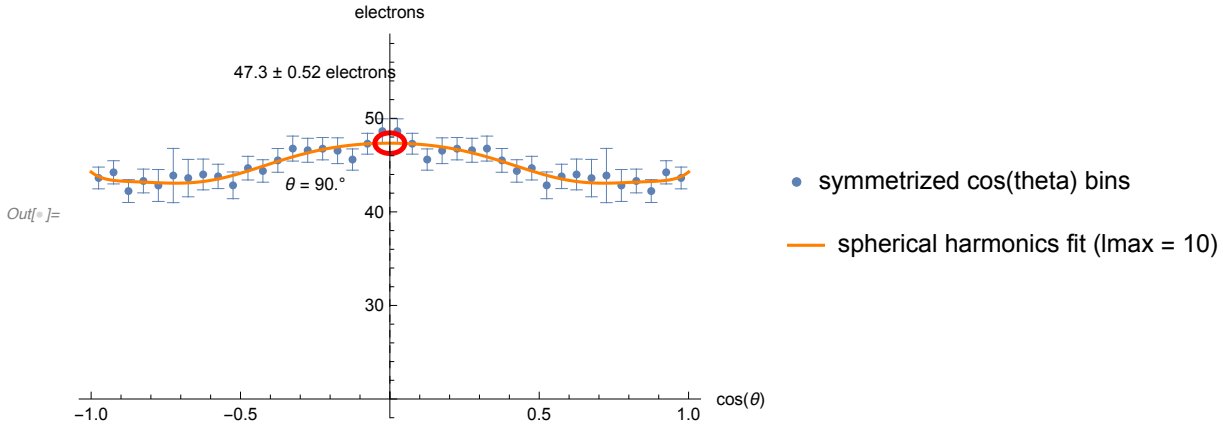
```
{45.12 ± 0.21, {{{{0.338099, 90}, {-0.962644, 19}, {-0.234118, 49}, {-0.608771, 79},
     {-0.302092, 61}, {0.428077, 38}, {-0.63474, 21}, {-0.786352, 26},
     {-0.811839, 34}, {0.378535, 49}, {0.752824, 27}, {-0.944244, 20},
     ... 4977 ... , {-0.464546, 19}, {0.5626, 39}, {-0.0596975, 40},
     {-0.438202, 33}, {-0.0369359, 67}, {0.045922, 79}, {0.339389, 71},
     {-0.572022, 35}, {-0.210192, 58}, {-0.157178, 32}, {-0.0474263, 36}}}}}
```

large output | show less | show more | show all | set si | ze limit...

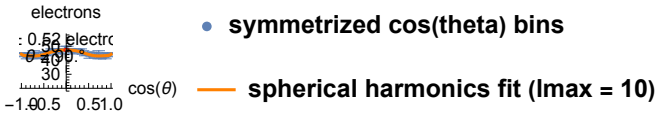
In[ ]:= `MakeElectronsPlot[r2[[2, 1, 1]]]`

$$\left\{\left\{-0.975, \frac{897}{20}\right\}, \left\{-0.925, \frac{2174}{49}\right\}, \left\{-0.875, \frac{1171}{28}\right\}, \left\{-0.825, \frac{4852}{113}\right\}, \left\{-0.775, \frac{3752}{81}\right\}, \left\{-0.725, \frac{1351}{30}\right\}, \left\{-0.675, \frac{1840}{43}\right\}, \left\{-0.625, \frac{2844}{65}\right\}, \left\{-0.575, \frac{2237}{51}\right\}, \left\{-0.525, \frac{1519}{36}\right\}, \left\{-0.475, \frac{5539}{124}\right\}, \left\{-0.425, \frac{6149}{137}\right\}, \left\{-0.375, \frac{7432}{163}\right\}, \left\{-0.325, \frac{6892}{149}\right\}, \left\{-0.275, \frac{7209}{155}\right\}, \left\{-0.225, \frac{6992}{147}\right\}, \left\{-0.175, \frac{6577}{143}\right\}, \left\{-0.125, \frac{7638}{163}\right\}, \left\{-0.075, \frac{8171}{176}\right\}, \left\{-0.025, \frac{3597}{73}\right\}, \left\{0.025, \frac{6957}{145}\right\}, \left\{0.075, \frac{7655}{159}\right\}, \left\{0.125, \frac{7403}{167}\right\}, \left\{0.175, \frac{7251}{154}\right\}, \left\{0.225, \frac{781}{17}\right\}, \left\{0.275, \frac{3033}{65}\right\}, \left\{0.325, \frac{6997}{148}\right\}, \left\{0.375, \frac{3179}{70}\right\}, \left\{0.425, \frac{6229}{142}\right\}, \left\{0.475, \frac{1832}{41}\right\}, \left\{0.525, \frac{1174}{27}\right\}, \left\{0.575, \frac{5378}{123}\right\}, \left\{0.625, \frac{3805}{86}\right\}, \left\{0.675, \frac{1644}{37}\right\}, \left\{0.725, \frac{983}{23}\right\}, \left\{0.775, \frac{3028}{77}\right\}, \left\{0.825, \frac{131}{3}\right\}, \left\{0.875, \frac{6607}{155}\right\}, \left\{0.925, \frac{6525}{148}\right\}, \left\{0.975, \frac{6318}{149}\right\}\right\}$$

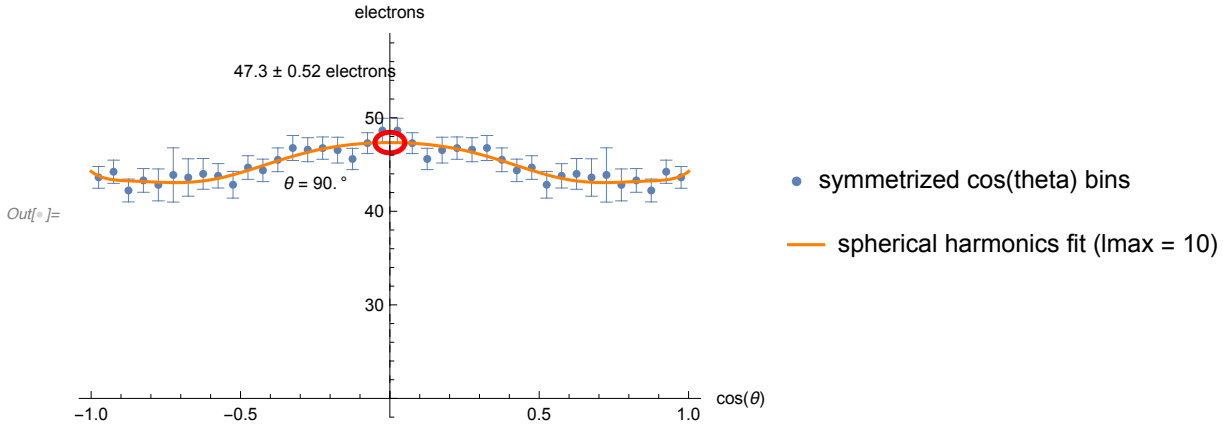
Out[ ]=



In[ ]:= `p2 =`



Out[ ]=



In[ ]:=
```
Export["/Users/wbowers/Documents/stem-fellowship/tif-figs/e-bins-1.TIFF",
 p1, ImageResolution → 600]
Export["/Users/wbowers/Documents/stem-fellowship/tif-figs/e-bins-2.TIFF",
 p2, ImageResolution → 600]
```

Out[ ]= /Users/wbowers/Documents/stem-fellowship/tif-figs/e-bins-1.TIFF

Out[ ]= /Users/wbowers/Documents/stem-fellowship/tif-figs/e-bins-2.TIFF

Using Spherical Harmonics to create max electrons function for sphere-in-cylinder

```
MaxElectrons[λ_, Ni_, reps_] :=
 Module[{idx, list, i, thetaslist, constants, SHYs, fit, max, error},
  list = Last[
     Last[Reap[AvgElectronsEulerSC[λ, Ni, 15, 2 / 5, 4 / 5, False, True, reps] ]]][[1]];
  fit = LinearModelFit[list,
    Table[SphericalHarmonicY[l, 0, theta, 0], {l, 0, lmax, 2}] /. Cos[theta] → z,
    z, IncludeConstantBasis → False];
  max = NMaximize[{fit[z], -1 ≤ z ≤ 1}, z];
  Print[z /. max[[2]]];
  SHYs =
   Table[SphericalHarmonicY[l, 0, theta, 0], {l, 0, lmax, 2}] /. Cos[theta] → z;
  error = Sqrt[SHYs.fit["CovarianceMatrix"].Transpose[SHYs]] /. max[[2]];
  Around[max[[1]], error]
 ]
```

In[ ]:=
```
MaxElectronsFAST[λ_, Ni_, reps_] :=
 Module[{idx, list, i, thetaslist, constants, SHYs, fit, max, error},
  list = Last[Last[
      Reap[AvgElectronsEulerSCFAST[λ, Ni, 15, 2 / 5, 4 / 5, False, True, reps] ]]][[1]];
  Clear[theta, z];
  fit = LinearModelFit[list,
    Table[SphericalHarmonicY[l, 0, theta, 0], {l, 0, lmax, 2}] /. Cos[theta] → z,
    z, IncludeConstantBasis → False];
  max = NMaximize[{fit[z], -1 ≤ z ≤ 1}, z];
  Print[z /. max[[2]]];
  SHYs =
   Table[SphericalHarmonicY[l, 0, theta, 0], {l, 0, lmax, 2}] /. Cos[theta] → z;
  error = Sqrt[SHYs.fit["CovarianceMatrix"].Transpose[SHYs]] /. max[[2]];
  Around[max[[1]], error]
 ]
```

In[ ]:= MaxElectronsFAST[53 / 100, 12.1, 500]

1.97348×10^{-32}

Out[ ]= 70. ± 4.

In[ ]:= MaxElectronsFAST[53 / 100, 12.1, 500]

1.03398×10^{-25}

Out[ ]= 69. ± 4.

Ensuring exponential growth of ions for the parallel plate case (1D AND 3D)

In[ ]:= t1 = Table[{Nc, AvgElectronsWeighted[Nc, Nc, 1000] }, {Nc, 0.2, 14, 0.8}]

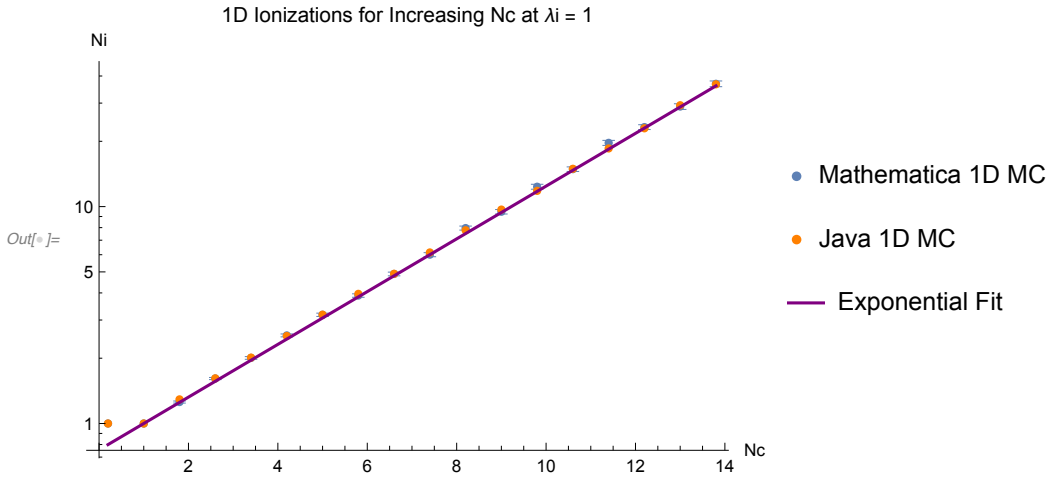
Out[ ]= {{0.2, 1}, {1., 1}, {1.8, 1.256 ± 0.014}, {2.6, 1.612 ± 0.018},
{3.4, 2.006 ± 0.029}, {4.2, 2.55 ± 0.04}, {5., 3.17 ± 0.06},
{5.8, 3.89 ± 0.07}, {6.6, 4.89 ± 0.10}, {7.4, 6.00 ± 0.13}, {8.2, 7.96 ± 0.17},
{9., 9.47 ± 0.22}, {9.8, 12.35 ± 0.32}, {10.6, 14.9 ± 0.4},
{11.4, 19.7 ± 0.5}, {12.2, 23.3 ± 0.6}, {13., 28.9 ± 0.9}, {13.8, 36.8 ± 1.1}}

In[ ]:= java1d = {{0.2, Around[1.0, 0.0]}, {1.0, Around[1.0, 0.0]}, {1.8, Around[1.292, 0.0]},
{2.6, Around[1.6152, 0.0]}, {3.4000000000000004, Around[2.0128, 0.0]},
{4.2, Around[2.5308, 0.0]}, {5.0, Around[3.1744, 0.0]}, {5.8, Around[3.96, 0.0]},
{6.6, Around[4.8992, 0.0]}, {7.3999999999999995, Around[6.146, 0.0]},
{8.2, Around[7.778, 0.0]}, {9.0, Around[9.6872, 0.0]},
{9.8, Around[11.818, 0.0]}, {10.600000000000001, Around[14.9372, 0.0]},
{11.400000000000002, Around[18.5404, 0.0]},
{12.200000000000003, Around[22.9588, 0.0]},
{13.000000000000004, Around[29.3304, 0.0]}, {13.800000000000004,
Around[36.6168, 0.0]}, {14.600000000000005, Around[45.6408, 0.0]}}

Out[ ]= {{0.2, 1.}, {1., 1.}, {1.8, 1.292}, {2.6, 1.6152}, {3.4, 2.0128},
{4.2, 2.5308}, {5., 3.1744}, {5.8, 3.96}, {6.6, 4.8992}, {7.4, 6.146},
{8.2, 7.778}, {9., 9.6872}, {9.8, 11.818}, {10.6, 14.9372}, {11.4, 18.5404},
{12.2, 22.9588}, {13., 29.3304}, {13.8, 36.6168}, {14.6, 45.6408}}

In[ ]:=
```
Show[ListLogPlot[t1, PlotLegends → {"Mathematica 1D MC"}],
 ListLogPlot[java1d, PlotStyle → Orange, PlotLegends → {"Java 1D MC"}],
 LogPlot[Exp[0.28 (x - 1)], {x, 0.2, 13.8},
  PlotStyle → Purple, PlotLegends → {"Exponential Fit"}],
 PlotLabel → "1D Ionizations for Increasing Nc at λi = 1", AxesLabel → {Nc, Ni}]
```

Out[ ]=



In[ ]:=
```
t2 = Table[{Nc, AvgElectrons3D[Nc, Nc, 5, 15, 0.8, 100]}, {Nc, 0.2, 14, 0.8}]
```

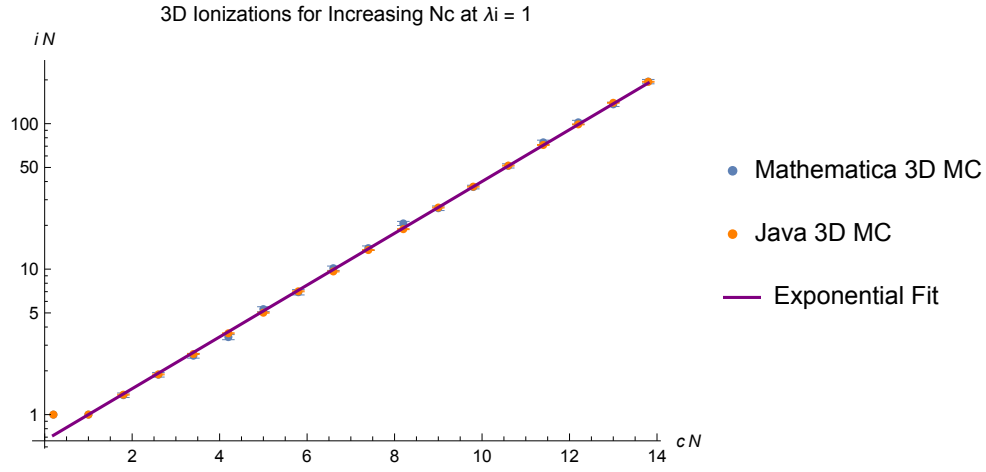
```
{{0.2, 1}, {1., 1}, {1.8, 1.36 ±0.05}, {2.6, 1.88 ±0.07}, {3.4, 2.54 ±0.10},
 {4.2, 3.41 ±0.14}, {5., 5.31 ±0.20}, {5.8, 6.94 ±0.29}, {6.6, 10.15 ±0.35},
 {7.4, 13.9 ±0.5}, {8.2, 20.6 ±0.7}, {9., 26.2 ±1.0}, {9.8, 36.7 ±1.1}, {10.6, 51.3 ±1.7},
 {11.4, 74.4 ±2.6}, {12.2, 102.1 ±3.3}, {13., 136. ±5.}, {13.8, 195. ±7.}}
```

In[ ]:=
```
java3d = {{0.2, Around[1.0, 0.0]},
  {1.0, Around[1.0, 0.0]}, {1.8, Around[1.3708, 0.009660380116744999]},
  {2.6, Around[1.8896, 0.014252183552003757]},
  {3.4000000000000004, Around[2.6092, 0.01951076994892846]},
  {4.2, Around[3.622, 0.02711690247797481]},
  {5.0, Around[5.0172, 0.03788458346082171]},
  {5.8, Around[7.038, 0.05230891319842165]},
  {6.6, Around[9.6412, 0.071414459488258]},
  {7.3999999999999995, Around[13.5272, 0.0970370241918007]},
  {8.2, Around[18.8308, 0.13820415530656086]},
  {9.0, Around[26.5088, 0.18488258172148112]},
  {9.8, Around[36.892, 0.25806691845333396]},
  {10.600000000000001, Around[51.484, 0.353070726059242]},
  {11.400000000000002, Around[71.3404, 0.49144933730344953]},
  {12.200000000000003, Around[98.8472, 0.6953129517447517]},
  {13.000000000000004, Around[139.1028, 0.961511358676537]},
  {13.800000000000004, Around[194.0804, 1.3399496611201482]}}
```

Out[ ]=
```
{{0.2, 1.}, {1., 1.}, {1.8, 1.371 ± 0.010}, {2.6, 1.890 ± 0.014},
 {3.4, 2.609 ± 0.020}, {4.2, 3.622 ± 0.027}, {5., 5.02 ± 0.04},
 {5.8, 7.04 ± 0.05}, {6.6, 9.64 ± 0.07}, {7.4, 13.53 ± 0.10}, {8.2, 18.83 ± 0.14},
 {9., 26.51 ± 0.18}, {9.8, 36.89 ± 0.26}, {10.6, 51.48 ± 0.35},
 {11.4, 71.3 ± 0.5}, {12.2, 98.8 ± 0.7}, {13., 139.1 ± 1.0}, {13.8, 194.1 ± 1.3}}
```

In[ ]:=
```
Show[ListLogPlot[t2, PlotLegends → {"Mathematica 3D MC"}],
 ListLogPlot[java3d, PlotStyle → Orange, PlotLegends → {"Java 3D MC"}],
 LogPlot[Exp[0.41 (x - 1)], {x, 0.2, 13.8},
  PlotStyle → Purple, PlotLegends → {"Exponential Fit"}],
 PlotLabel → "3D Ionizations for Increasing Nc at λi = 1", AxesLabel → {N c, N i}]
```

Out[ ]=

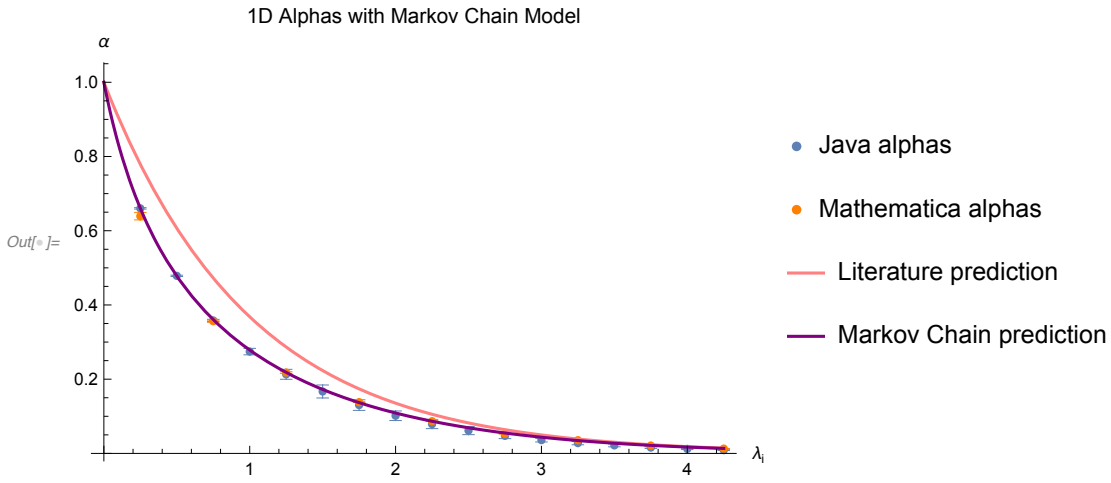


Exploring 1D alphas

```
In[ ]:= FindAlpha[λi_, reps_] := Module[{electrons, result,
          length, i, total, slopes, alpha, e, fittedmodel, a, b, c, d, params},
        electrons = Table[
          {Nc, Log[AvgElectronsWeighted[Nc, ((1 / λi) Nc), reps]]}, {Nc, λi + 5, 40, 4}];
        fittedmodel = NonlinearModelFit[
          {#[[1]], #[[2, 1]]} & /@ electrons, a * x + b + c Exp[-d x], {a, b, c, d}, x];
        params = fittedmodel["ParameterTable"][[1, 1]];
        {Around[params[[2, 2]], params[[2, 3]]], Around[params[[3, 2]], params[[3, 3]]]}]

In[ ]:= markovmodel[x_] := 1 / x * ProductLog[x Exp[-x]]

In[ ]:= javaalphas = {{0.25, Around[0.6602595438326849, 0.0019041936606251 15]},
          {0.5, Around[0.47844948673235005, 0.0017481441536300 99]},
          {0.75, Around[0.35821274297950323, 0.0031233087246661244]},
          {1.0, Around[0.27433365451618424, 0.0086911893482466 31]},
          {1.25, Around[0.21314895885656557, 0.0133893175558122 27]},
          {1.5, Around[0.1669038047699682, 0.01749867647681158]},
          {1.75, Around[0.13018046550569878, 0.014267346022973922]},
          {2.0, Around[0.1015082557377325, 0.012894206875325626]},
          {2.2500000000000004, Around[0.078742597634773, 0.011952013460590393]},
          {2.5, Around[0.061520625296323746, 0.010962062286845231]},
          {2.75, Around[0.04725434577446056, 0.00734274289001424]},
          {3.0, Around[0.0358071249036958, 0.004959892432166247]},
          {3.25, Around[0.02778741774482209, 0.004504199943380026]},
          {3.5, Around[0.021572007247372436, 0.003796229153906858]},
          {3.7500000000000004, Around[0.015726014436025514, 0.0022654561964774527]},
          {4.0, Around[0.012178682642516597, 0.0017503399885901638]},
          {4.25, Around[0.008830658060106817, 0.0010123646555449694]}}

Out[ ]= {{0.25, 0.6603 ± 0.0019}, {0.5, 0.4784 ± 0.0017}, {0.75, 0.3582 ± 0.0031},
         {1., 0.274 ± 0.009}, {1.25, 0.213 ± 0.013}, {1.5, 0.167 ± 0.017},
         {1.75, 0.130 ± 0.014}, {2., 0.102 ± 0.013}, {2.25, 0.079 ± 0.012}, {2.5, 0.062 ± 0.011},
         {2.75, 0.047 ± 0.007}, {3., 0.036 ± 0.005}, {3.25, 0.028 ± 0.005}, {3.5, 0.022 ± 0.004},
         {3.75, 0.0157 ± 0.0023}, {4., 0.0122 ± 0.0018}, {4.25, 0.0088 ± 0.0010}}

In[ ]:= alphas1d = Table[{r, FindAlpha[r, 1000][[1]]}, {r, 0.25, 4.25, 0.5}]

Out[ ]= {{0.25, 0.639 ± 0.010}, {0.75, 0.3563 ± 0.0028}, {1.25, 0.2179 ± 0.0022},
         {1.75, 0.1382 ± 0.0009}, {2.25, 0.0868 ± 0.0009}, {2.75, 0.0529 ± 0.0011},
         {3.25, 0.0356 ± 0.0018}, {3.75, 0.0212 ± 0.0006}, {4.25, 0.0129 ± 0.0006}}
```

In[ ]:=
```
fig8 = Show[ListPlot[javaalphas, PlotLegends → {"Java alphas"}],
  ListPlot[alphas1d, PlotStyle → Orange, PlotLegends → {"Mathematica alphas"}],
  Plot[Exp[-x], {x, 0, 4.25}, PlotLegends → {"Literature prediction"},
   PlotStyle → Pink], Plot[1 / x * ProductLog[x Exp[-x]], {x, 0, 4.25},
   PlotStyle → Purple, PlotLegends → {"Markov Chain prediction"}],
  PlotRange → {0, 1}, AxesLabel → {Subscript["λ", "i"], "α"},
  PlotLabel → "1D Alphas with Markov Chain Model"]
```

Out[ ]=

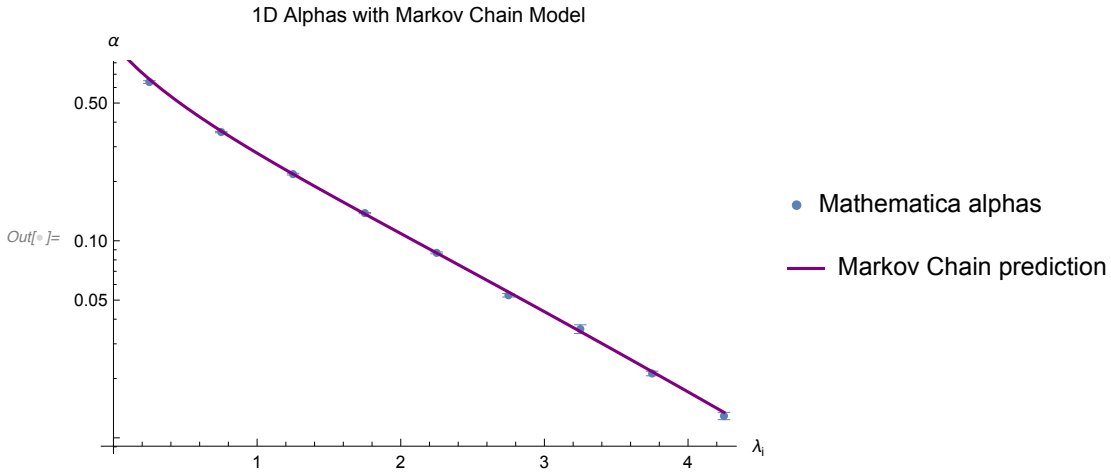


In[ ]:=
```
Export["/Users/wbowers/Documents/stem-fellowship/figures/alphas-plot.TIFF",
 fig8, ImageResolution → 600]
```

Out[ ]=
```
/Users/wbowers/Documents/stem-fellowship/figures/alphas-plot.TIFF
```

In[ ]:=
```
Show[ListLogPlot[alphas1d, PlotLegends → {"Mathematica alphas"}],
 LogPlot[1 / x * ProductLog[x Exp[-x]], {x, 0, 4.25}, PlotStyle → Purple,
  PlotLegends → {"Markov Chain prediction"}], AxesLabel → {Subscript["λ", "i"], "α"},
 PlotLabel → "1D Alphas with Markov Chain Model"]
```

Out[ ]=



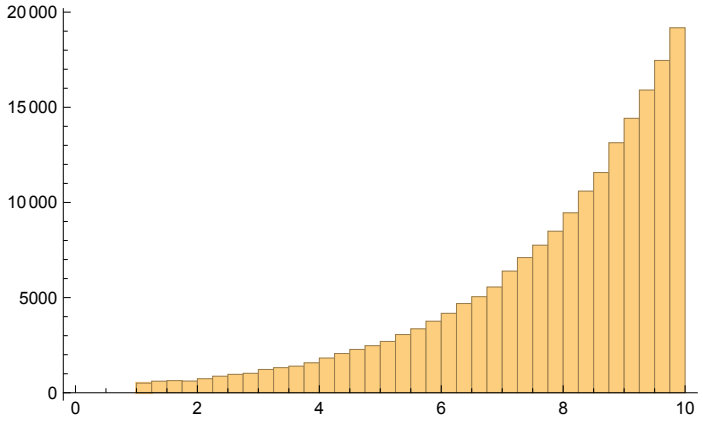
In[ ]:=
```
l1 = Reap[AvgElectrons3D[10, 10, 5000]]; l1[[1]]
```

Out[ ]=
```
39.79 ± 0.19
```

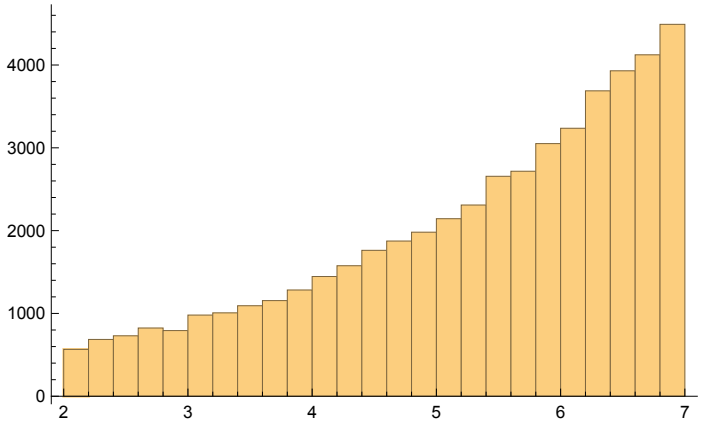
Examining the sample space for the alpha calculation

In[ ]:=
```
Histogram[l1[[2, 1]], {0, 10, 0.25}]
Histogram[l1[[2, 1]], {2, 7, 0.2}]
```

Out[ ]=

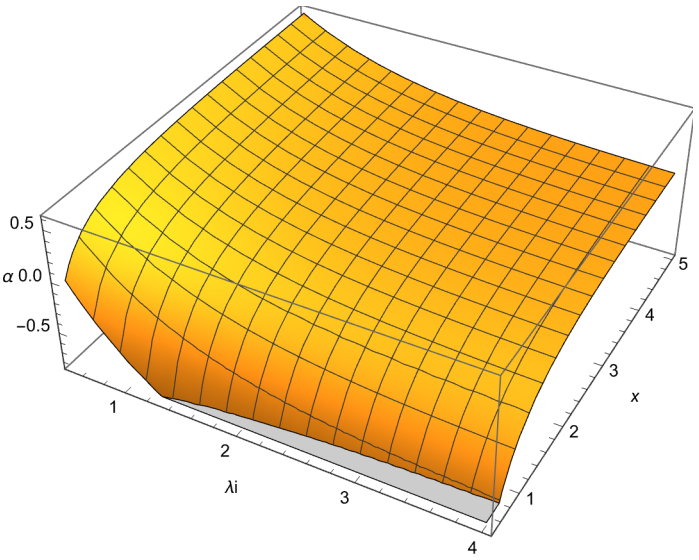


Out[ ]=



In[ ]:=
```
Clear[α]
FindAlphav2[λi_, x_] := (Exp[-λi] - Exp[-x]) / (1 - Exp[-x])
Plot3D[FindAlphav2[λi, x], {λi, 1 / 2, 4}, {x, 1 / 2, 5}, AxesLabel → {λi, x, α}]
```

Out[ ]=




```
FindRoot[FindAlphav2[1 / 2, x] == FindAlpha[1 / 2][[1]], {x, 1, 6}]
FindRoot[FindAlphav2[1, x] == FindAlpha[1][[1]], {x, 1, 6}]
FindRoot[FindAlphav2[4, x] == FindAlpha[4][[1]], {x, 1, 6}]
```

Out[ ]= {x → 1.42628}

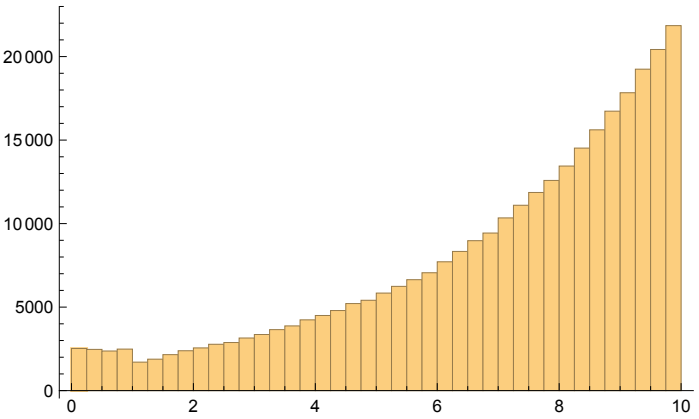
Out[ ]= {x → 2.16402}

Out[ ]= {x → 6.78437}

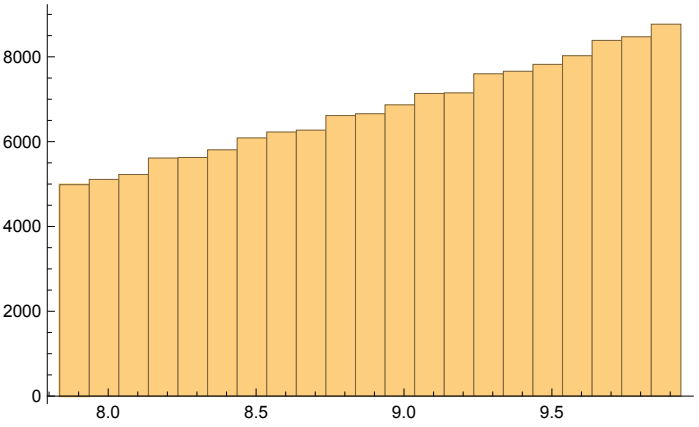
```
l1 = Reap[AvgElectronsBanked[10, 10, 10 000]]; l1[[1]]
Histogram[l1[[2, 1]], {0, 10, 0.25}]
Histogram[l1[[2, 1]], {(10 - 2.16401), 10, 0.1}]
```

Out[ ]= 12.65 ± 0.06

Out[ ]=



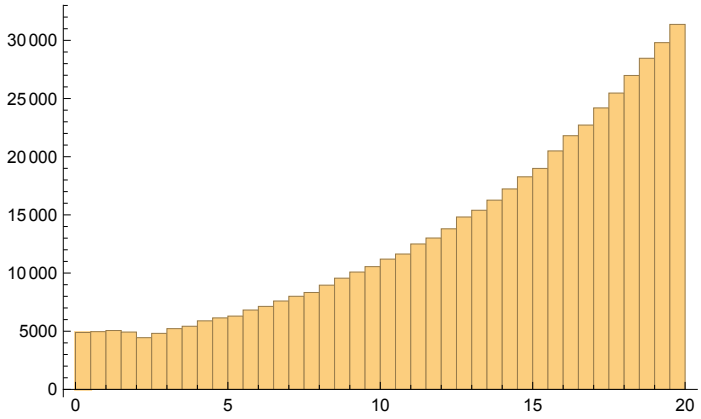
Out[ ]=



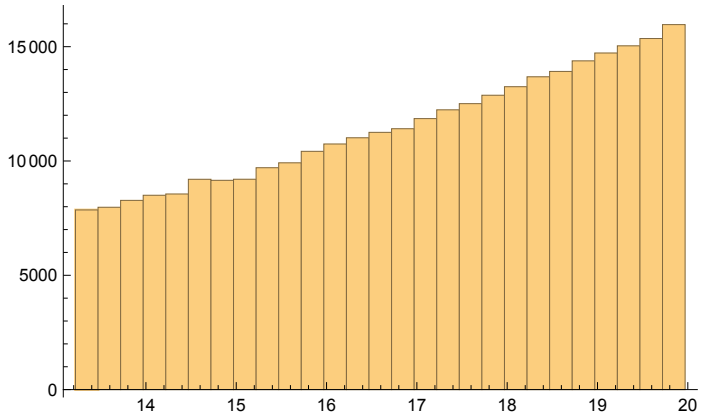
```
l2 = Reap[AvgElectronsBanked[20, 10, 10 000]]; l2[[1]]
Histogram[l2[[2, 1]], {0, 20, 0.5}]
Histogram[l2[[2, 1]], {20 - 6.78, 20, 0.25}]
```

Out[ ]= 7.25 ± 0.04

Out[ ]=



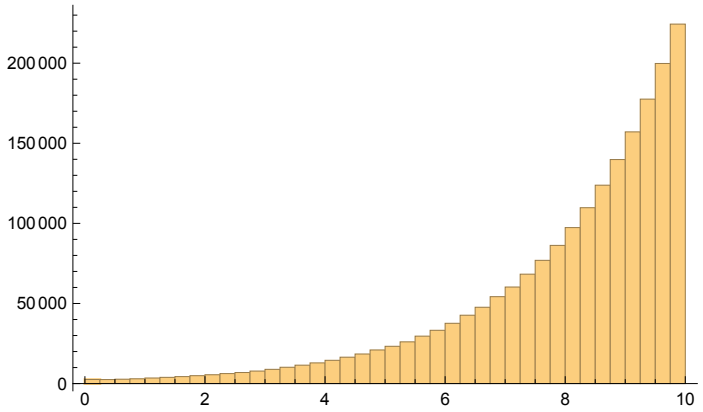
Out[ ]=



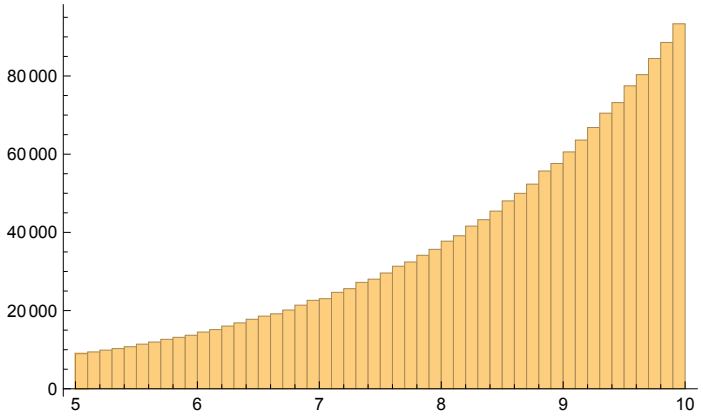
```
l3 = Reap[AvgElectronsBanked[10, 20, 10 000]]; l3[[1]]
Histogram[l3[[2, 1]], {0, 10, 0.25}]
Histogram[l3[[2, 1]], {10 - 5, 10, 0.1}]
```

Out[ ]= 95.4 ± 0.5

Out[ ]=

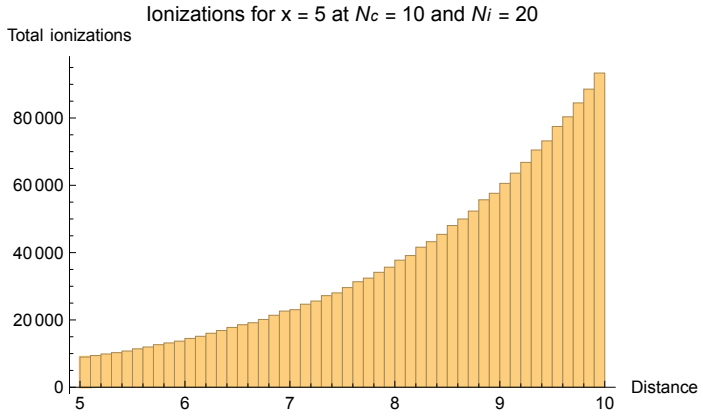


Out[ ]=

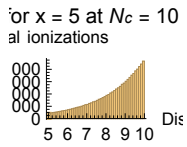


In[ ]:=
```
Histogram[l3[[2, 1]], {10 - 5, 10, 0.1}, AxesLabel → {"Distance", "Total ionizations"},
 PlotLabel → "Ionizations for x = 5 at Nc = 10 and Ni = 20"]
```

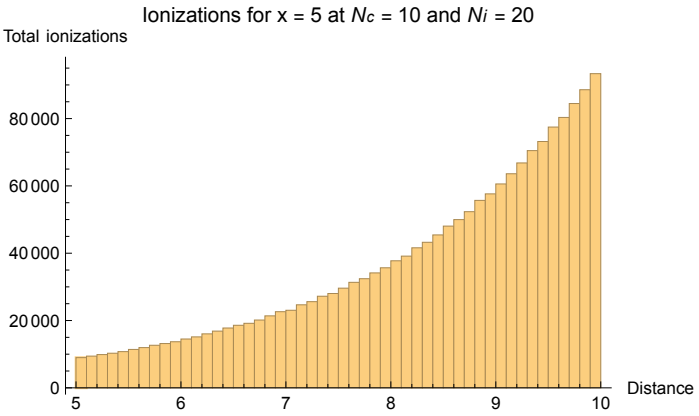
Out[ ]=



In[ ]:=
```
fig9 =
```



Out[ ]=



In[ ]:=
```
Export["/Users/wbowers/Documents/stem-fellowship/tif-figs/sample-space.TIFF",
 fig9, ImageResolution → 600]
```

Out[ ]=
```
/Users/wbowers/Documents/stem-fellowship/tif-figs/sample-space.TIFF
```

Parallel plate breakdown 1D

In[ ]:=
```
FindBreakdown[r_, ɣ_, reps_] := Module[{alpha, idx, Nc, Ni},
alpha = FindAlpha[r, reps][[1]];
Nc = Log[ɣ] / alpha + r ;
Ni = (1 / r) (Nc);
{Nc, Ni}]
```

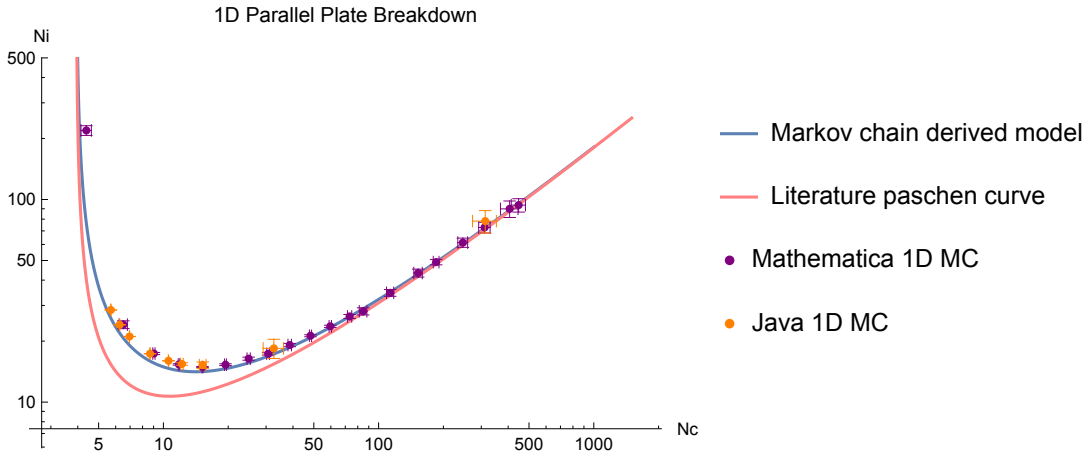
In[ ]:=
```
bdpp1d = Table[FindBreakdown[r, 50, 500], {r, 1 / 50, 5, 1 / 4}]
```

Out[ ]=
```
{{4.38 ±0.24, 219. ±12.}, {6.51 ±0.29, 24.1 ±1.1}, {9.03 ±0.11, 17.37 ±0.21},
 {11.81 ±0.10, 15.34 ±0.13}, {15.19 ±0.09, 14.90 ±0.09}, {19.43 ±0.20, 15.30 ±0.15},
 {24.9 ±0.4, 16.38 ±0.29}, {30.7 ±0.5, 17.33 ±0.27}, {38.6 ±0.7, 19.1 ±0.4},
 {48.3 ±0.6, 21.26 ±0.25}, {59.6 ±0.8, 23.67 ±0.30}, {73.3 ±1.7, 26.5 ±0.6},
 {84.8 ±3.5, 28.1 ±1.1}, {113. ±4., 34.6 ±1.3}, {153. ±7., 43.4 ±2.0}, {185. ±6., 49.1 ±1.5},
 {246. ±13., 61.3 ±3.2}, {311. ±19., 73. ±4.}, {406. ±38., 90. ±8.}, {447. ±34., 94. ±7.}}
```

```
In[•]:= javabd1d = {{Around[5.700231943597744, 0.01207241166528697],
        Around[28.50115971798872, 0.06036205832643485]},
       {Around[6.259659518165803, 0.01294585426723786],
        Around[24.075613531406933, 0.049791747181684075]},
       {Around[6.956648887715002, 0.006510191038770662],
        Around[21.08075420519698, 0.01972785163263837]},
       {Around[8.666186577739973, 0.01778912290701574],
        Around[17.332373155479946, 0.03557824581403148]},
       {Around[10.558215754398047, 0.018609784723081196],
        Around[15.997296597572799, 0.028196643519819993]},
       {Around[12.189849737655585, 0.1691424999247856],
        Around[15.430189541336182, 0.2141044302845387]},
       {Around[15.233034583855726, 0.5468246433362867],
        Around[15.233034583855726, 0.5468246433362867]},
       {Around[32.607032745474434, 3.5391212217852397],
        Around[18.422052398573125, 1.9995035151329037]},
       {Around[37.56666730405342, 4.788665312298926], Around[19.56597255419449,
          2.494096516822358]} {Around[226.1883191276727, 34.15818691413011],
         Around[61.13197814261424, 9.231942409224354]}, {Around[312.9949279966182,
         39.66595313492955], Around[78.24873199915454, 9.916488283732388]}}

Out[•]= {{5.700 ± 0.012, 28.50 ± 0.06}, {6.260 ± 0.013, 24.08 ± 0.05},
 {6.957 ± 0.007, 21.081 ± 0.020}, {8.666 ± 0.018, 17.33 ± 0.04},
 {10.558 ± 0.019, 15.997 ± 0.028}, {12.19 ± 0.17, 15.43 ± 0.21}, {15.2 ± 0.5, 15.2 ± 0.5},
 {32.6 ± 3.5, 18.4 ± 2.0}, {(8.5 ± 1.7) × 10^3, 1196. ± 236.}, {313. ± 40., 78. ± 10.}}

In[•]:= Solve[Log[51] == Ni * ProductLog[Nc / Ni * Exp[-Nc / Ni]], Ni]

Out[•]= {{Ni → (-Nc - Log[51]) / Log[Log[51] / Nc]}}
```

In[ ]:=
```
markovmodelfig = Show[LogLogPlot[(-Nc - Log[51])/Log[Log[51]/Nc],
   {Nc, 3, 1000}, PlotLegends → {"Markov chain derived model"}],
  LogLogPlot[Nc / (Log[Nc] - Log[Log[51]]), {Nc, 3, 1500},
   PlotStyle → Pink, PlotLegends → {"Literature paschen curve"}],
  ListLogLogPlot[bdpp1d, PlotLegends → {"Mathematica 1D MC"}, PlotStyle → Purple],
  ListLogLogPlot[javabd1d, PlotStyle → Orange, PlotLegends → {"Java 1D MC"}],
  AxesLabel → {"Nc", "Ni"}, PlotLabel → "1D Parallel Plate Breakdown",
  PlotRange → {{1, 7.5}, {2, 6}}, AxesOrigin → {1, 2}]
```

Out[ ]=



In[ ]:=
```
Export["/Users/wbowers/Documents/stem-fellowship/figures/1d-pp-breakdown.TIFF",
 markovmodelfig, ImageResolution → 600]
```

Out[ ]=
```
/Users/wbowers/Documents/stem-fellowship/figures/1d-pp-breakdown.TIFF
```

Parallel plate breakdown 3D

```
FindBreakdown3D[d_, ɣ_, tbound_, Ni0_, reps_] :=
 Module[{region, list, i, errors, max, params, fit,
   error, p1, p2, p3, p4, f, vals, errorslist, listForFit},
  list =
   Table[{Ni, AvgElectrons3D[d, Ni, d, 15, tbound, reps] }, {Ni, Ni0 - 3, Ni0 + 3}];
  listForFit = {#[[1]], #[[2, 1]]} & /@ list;
  errorslist = #[[2, 2]] & /@ list;
  fit = NonlinearModelFit[listForFit,
    p1 Exp[p2 x] + p3, {p1, p2, p3}, x, Weights → 1 / errorslist^2];
  f = (Log[(ɣ - p3) / p1] / p2);
  error = {D[f, p1], D[f, p2], D[f, p3]}.fit["CovarianceMatrix"].
     Transpose[{D[f, p1], D[f, p2], D[f, p3]}] /. fit["BestFitParameters"];
  Around[f, Sqrt[error]] /. fit["BestFitParameters"]
 ]
```

```
In[•]:= FindBreakdown3Dv2[4.5, 50, 0.8, 30, 3000]
```

```
{{27, 45.5 ± 0.7}, {28, 46.1 ± 0.7}, {29, 47.5 ± 0.7},
 {30, 48.5 ± 0.7}, {31, 48.7 ± 0.8}, {32, 49.9 ± 0.8}, {33, 51.3 ± 0.8}}
```

```
Out[•]= 32.01 ± 0.21
```

```
In[•]:= left = Table[{d, FindBreakdown3Dv2[d, 50, 0.8, 20, 500]}, {d, 5, 6, 0.5}]
```

```
Out[•]= {{5., 21.2 ± 0.4}, {5.5, 16.75 ± 0.35}, {6., 11. ± 5.}}
```

```
In[•]:= min = Table[{d, FindBreakdown3Dv2[d, 50, 0.8, 10, 100]}, {d, 6, 16.5, 2}]
```

```
Out[•]= {{6, 14.4 ± 0.4}, {8, 11.73 ± 0.13}, {10, 10.790 ± 0.034},
 {12, 10.60 ± 0.04}, {14, 10.52 ± 0.07}, {16, 10.70 ± 0.05}}
```

```
In[•]:= right2 = Table[{d, FindBreakdown3Dv2[d, 50, 0.8, 13, 50]}, {d, 17, 57, 4}]
```

```
Out[•]= {{17, 10.81 ± 0.08}, {21, 11.24 ± 0.04}, {25, 12.07 ± 0.09},
 {29, 12.66 ± 0.05}, {33, 13.45 ± 0.05}, {37, 14.255 ± 0.029}, {41, 14.939 ± 0.018},
 {45, 15.66 ± 0.09}, {49, 16.49 ± 0.22}, {53, 17.48 ± 0.09}, {57, 17.74 ± 0.30}}
```

```
In[•]:= right3 = Table[{d, FindBreakdown3Dv2[d, 50, 0.8, 17, 25]}, {d, 60, 100, 8}]
```

```
Out[•]= {{60, 18.52 ± 0.09}, {68, 19.97 ± 0.07}, {76, 21.31 ± 0.35},
 {84, 23.7 ± 1.2}, {92, 24.7 ± 0.9}, {100, 26.2 ± 1.4}}
```

```
In[•]:= bdfinal = {{4.5, 32.01 ± 0.21}, {5.`, 21.2 ± 0.4},
   {5.5`, 16.75 ± 0.35}, {6, 14.4 ± 0.4}, {8, 11.73 ± 0.13}, {10, 10.790 ± 0.034},
   {12, 10.60 ± 0.04}, {14, 10.52 ± 0.07}, {16, 10.70 ± 0.05}, {17, 10.77 ± 0.06},
   {20, 11.18 ± 0.06}, {23, 11.64 ± 0.07}, {26, 12.21 ± 0.06}, {32, 13.27 ± 0.05},
   {37, 14.255 ± 0.029}, {41, 14.939 ± 0.018}, {49, 16.49 ± 0.22},
   {60, 18.52 ± 0.09}, {76, 21.31 ± 0.35}, {92, 24.7 ± 0.9}, {100, 26.2 ± 1.4}}
```

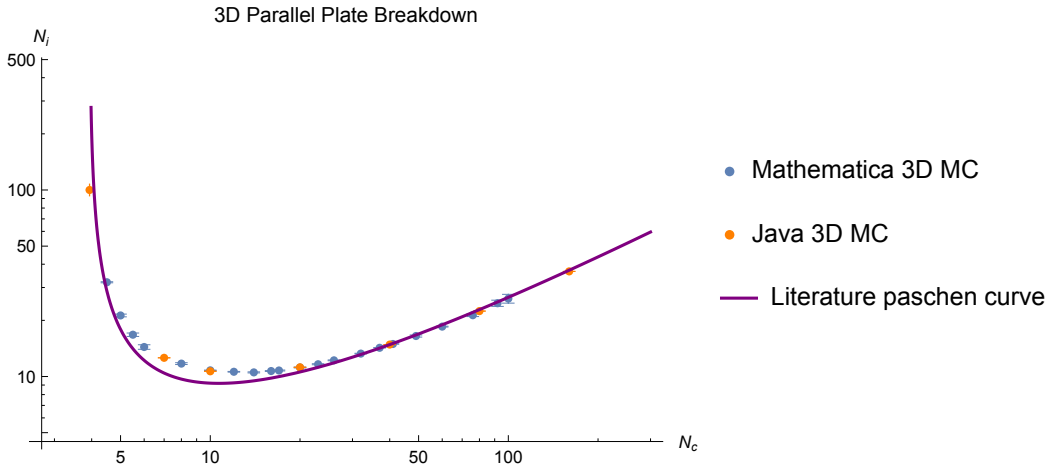
```
Out[•]= {{4.5, 32.01 ± 0.21}, {5., 21.2 ± 0.4}, {5.5, 16.75 ± 0.35}, {6, 14.4 ± 0.4}, {8, 11.73 ± 0.13},
 {10, 10.790 ± 0.034}, {12, 10.60 ± 0.04}, {14, 10.52 ± 0.07}, {16, 10.70 ± 0.05},
 {17, 10.77 ± 0.06}, {20, 11.18 ± 0.06}, {23, 11.64 ± 0.07}, {26, 12.21 ± 0.06},
 {32, 13.27 ± 0.05}, {37, 14.255 ± 0.029}, {41, 14.939 ± 0.018}, {49, 16.49 ± 0.22},
 {60, 18.52 ± 0.09}, {76, 21.31 ± 0.35}, {92, 24.7 ± 0.9}, {100, 26.2 ± 1.4}}
```

```
In[•]:= javabd3d = {{Around[3.9349346617986107`, 0.004098654929587719], 100},
   {7., Around[12.5973499575545, 0.021042290566655936`]},
   {10., Around[10.665900655226146`, 0.01147892717515875]},
   {20., Around[11.199362729457336`, 0.0055071173430502485`]},
   {40., Around[14.831063909226039`, 0.011373607275817859`]},
   {80., Around[22.424620315119526`, 0.003938560064511855]},
   {160., Around[36.60786133347448, 0.009721595486457082]}}
```

```
Out[•]= {{3.935 ± 0.004, 100}, {7., 12.597 ± 0.021}, {10., 10.666 ± 0.011}, {20., 11.199 ± 0.006},
 {40., 14.831 ± 0.011}, {80., 22.425 ± 0.004}, {160., 36.608 ± 0.010}}
```

In[ ]:=
```
fig10 = Show[ListLogLogPlot[bdfinal, PlotLegends → {"Mathematica 3D MC"}],
  ListLogLogPlot[javabd3d, PlotStyle → Orange, PlotLegends → {"Java 3D MC"}],
  LogLogPlot[0.86 * Nc / (Log[Nc] - Log[Log[51]]), {Nc, 3.9, 300}, PlotStyle → Purple,
   PlotLegends → {"Literature paschen curve"}], PlotRange → {1.5, 6},
  AxesOrigin → {1, 1.5}, AxesLabel → {Subscript[N, c], Subscript[N, i]},
  PlotLabel → "3D Parallel Plate Breakdown"]
```

Out[ ]=



In[ ]:=
```
Export["/Users/wbowers/Documents/stem-fellowship/figures/3d-pp-breakdown.TIFF",
 fig10, ImageResolution → 600]
```

Out[ ]=
```
/Users/wbowers/Documents/stem-fellowship/figures/3d-pp-breakdown.TIFF
```

Sphere-in-sphere breakdown

In[ ]:=
```
FindBreakdownSS[Nc_, γ_, Ni0_, dt_, reps_] :=
 Module[{region, list, i, errors, max, params, fit,
   error, p1, p2, p3, p4, f, vals, errorslist, listForFit},
  list = Table[{Ni, AvgElectronsEuler[Nc, Ni, 5, 15, dt, 4 / 5, False, reps]},
    {Ni, Ni0 - 3, Ni0 + 2}];
  Print[list];
  listForFit = {#[[1]], #[[2, 1]]} & /@ list;
  errorslist = #[[2, 2]] & /@ list;
  fit = NonlinearModelFit[listForFit,
    p1 Exp[p2 x] + p3, {p1, p2, p3}, x, Weights → 1 / errorslist^2];
  f = (Log[(γ - p3) / p1] / p2);
  error = {D[f, p1], D[f, p2], D[f, p3]}.fit["CovarianceMatrix"].
     Transpose[{D[f, p1], D[f, p2], D[f, p3]}] /. fit["BestFitParameters"];
  Around[f, Sqrt[error]] /. fit["BestFitParameters"]
 ]
```

```
In[•]:= FindNC[Ni_, γ_, Nc0_, dt_, reps_] :=
 Module[{region, list, i, errors, max, params, fit, error,
   p1, p2, p3, p4, f, vals, errorslist, listForFit},
  list = Table[{Nc, AvgElectronsEuler[Nc, Ni, 5, 15, dt, 4 / 5, False, reps]},
    {Nc, Nc0 - 1, Nc0 + 1, 1 / 2}];
  Print[list];
  listForFit = {#[[1]], #[[2, 1]]} & /@ list;
  errorslist = #[[2, 2]] & /@ list;
  fit = NonlinearModelFit[listForFit,
    p1 Exp[p2 x] + p3, {p1, p2, p3}, x, Weights → 1 / errorslist^2];
  f = (Log[(γ - p3) / p1] / p2);
  error = {D[f, p1], D[f, p2], D[f, p3]}.fit["CovarianceMatrix"].
     Transpose[{D[f, p1], D[f, p2], D[f, p3]}] /. fit["BestFitParameters"];
  Around[f, Sqrt[error]] /. fit["BestFitParameters"]
 ]

In[•]:= FindNC[100, 50, 4, 1 / 5, 300]

{{3, 18.9 ±1.2}, {7/2, 34.4 ±1.9}, {4, 50.1 ±2.8}, {9/2, 77. ±4.}, {5, 130. ±7.}}

Out[•]= 3.97 ±0.05

In[•]:= FindBreakdownSS[5, 50, 25, 2 / 5, 300]

{{22, 47.0 ±2.2}, {23, 44.0 ±2.2}, {24, 45.9 ±2.5},
 {25, 50.3 ±2.3}, {26, 53.8 ±2.6}, {27, 50.6 ±2.8}, {28, 58.9 ±3.2}}

Out[•]= 25.6 ±0.8

In[•]:= FindBreakdownSS[7, 50, 15, 2 / 5, 50]

{{12, 38.2 ±3.4}, {13, 45. ±4.}, {14, 53. ±6.},
 {15, 58. ±6.}, {16, 81. ±8.}, {17, 73. ±9.}, {18, 93. ±8.}}

Out[•]= 13.58 ±0.31

In[•]:= FindBreakdownSS[12, 50, 8, 2 / 5, 50]

{{5, 4.92 ±0.18}, {6, 7.76 ±0.29}, {7, 12.6 ±0.5}, {8, 18.2 ±0.7}, {9, 25.8 ±1.2}, {10, 40.9 ±2.5}}

Out[•]= 10.65 ±0.15

In[•]:= FindBreakdownSS[30, 50, 13, 2 / 5, 30]

{{10, 16.9 ±1.1}, {11, 27.4 ±1.5}, {12, 43.0 ±2.8}, {13, 66. ±4.}, {14, 99. ±6.}, {15, 160. ±9.}}

Out[•]= 12.36 ±0.04

In[•]:= FindBreakdownSS[50, 50, 15, 2 / 5, 20]

{{12, 13.4 ±0.9}, {13, 21.9 ±1.6}, {14, 31.3 ±2.2}, {15, 59. ±4.}, {16, 87. ±5.}, {17, 129. ±11.}}

Out[•]= 14.83 ±0.09
```

In[ ]:=
```
FindBreakdownSS[90, 50, 19, 2 / 5, 20]
```

{{16, 13.4 ±1.4}, {17, 21.4 ±1.8}, {18, 36.9 ±2.3}, {19, 45.6 ±3.0}, {20, 71. ±5.}, {21, 110. ±8.}}

Out[ ]= 19.03 ±0.12

In[ ]:=
```
FindBreakdownSS[150, 50, 24, 2 / 5, 10]
```

{{21, 14.8 ±2.5}, {22, 20.5 ±1.5}, {23, 23. ±5.}, {24, 37. ±5.}, {25, 42. ±7.}, {26, 81. ±11.}}

Out[ ]= 25.00 ±0.20

In[ ]:=
```
ssbd = {{4.036 ±0.018, 100}, {5, 25.6 ±0.8}, {7, 13.58 ±0.31}, {12, 10.65 ±0.15},
  {30, 12.36 ±0.04}, {50, 14.83 ±0.09}, {90, 19.03 ±0.12}, {150, 25.00 ±0.20}}
```

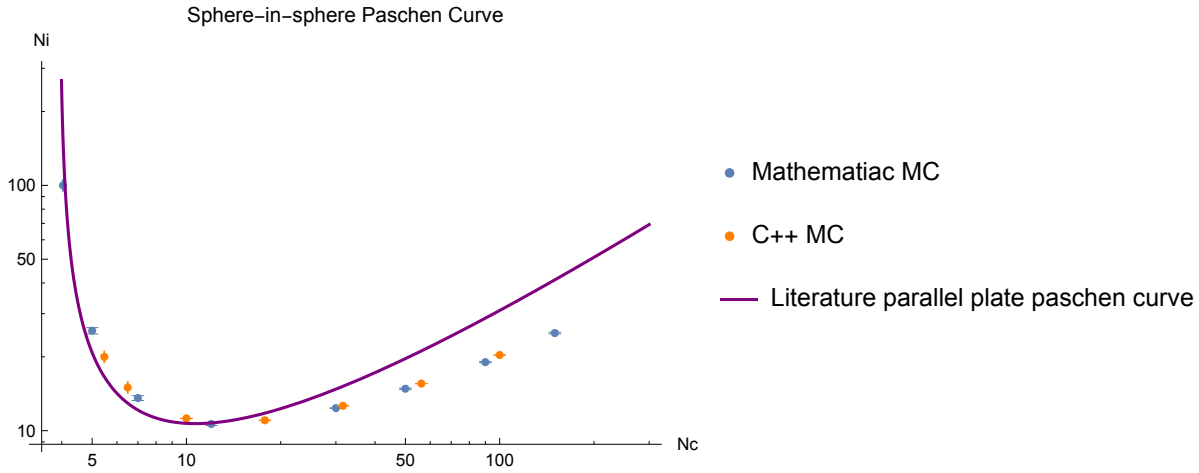
Out[ ]= {{4.036 ±0.018, 100}, {5, 25.6 ±0.8}, {7, 13.58 ±0.31}, {12, 10.65 ±0.15},
{30, 12.36 ±0.04}, {50, 14.83 ±0.09}, {90, 19.03 ±0.12}, {150, 25.00 ±0.20}}

In[ ]:=
```
drwhitmerdata =
 {{6.494 ±0.019, 15.`}, {5.469 ±0.017, 20.`}, {10, 11.227 ±0.008}, {17.8`, 11.026 ±0.009},
  {31.6`, 12.636 ±0.006}, {56.2`, 15.558 ±0.020}, {100.`, 20.350 ±0.025}}
```

Out[ ]= {{6.494 ±0.019, 15.}, {5.469 ±0.017, 20.}, {10, 11.227 ±0.008}, {17.8, 11.026 ±0.009},
{31.6, 12.636 ±0.006}, {56.2, 15.558 ±0.020}, {100., 20.350 ±0.025}}

In[ ]:=
```
fig11 = Show[ListLogLogPlot[ssbd, PlotLegends → {"Mathematiac MC"}],
  ListLogLogPlot[drwhitmerdata, PlotStyle → Orange, PlotLegends → {"C++ MC"}],
  LogLogPlot[Nc / (Log[Nc] - Log[Log[51]]), {Nc, 0, 300}, PlotStyle → Purple,
   PlotLegends → {"Literature parallel plate paschen curve"}], AxesLabel → {Nc, Ni},
  PlotLabel → "Sphere-in-sphere Paschen Curve", PlotRange → All]
```

Out[ ]=



In[ ]:=
```
Export["/Users/wbowers/Documents/High-res-fusor-figures/ss-breakdown.png",
 fig11, ImageResolution → 300]
```

Out[ ]= /Users/wbowers/Documents/High-res-fusor-figures/ss-breakdown.png

Sphere-in-cylinder breakdown

In[ ]:=
```
FindBreakdownNISC[λ_, ϒ_, Ni0_, reps_] :=
 Module[{region, list, i, errors, max, params, fit,
   error, p1, p2, p3, p4, f, vals, errorslist, listForFit},
  list =
   ParallelTable[{Ni, MaxElectrons[λ, Ni, reps]}, {Ni, Ni0 - 3 / 2, Ni0 + 3 / 2, 1 / 2}];
  Print[list];
  listForFit = {#[[1]], #[[2, 1]]} & /@ list;
  errorslist = #[[2, 2]] & /@ list;
  fit = NonlinearModelFit[listForFit,
    p1 Exp[p2 x] + p3, {p1, p2, p3}, x, Weights → 1 / errorslist^2];
  f = (Log[(ϒ - p3) / p1] / p2);
  error = {D[f, p1], D[f, p2], D[f, p3]}.fit["CovarianceMatrix"].
     Transpose[{D[f, p1], D[f, p2], D[f, p3]}] /. fit["BestFitParameters"];
  Around[f, Sqrt[error]] /. fit["BestFitParameters"]
 ]
```

In[ ]:=
```
FindBreakdownNCSC[Ni_, ϒ_, Nc0_, reps_] :=
 Module[{region, list, i, errors, max, params, fit,
   error, p1, p2, p3, p4, f, vals, errorslist, listForFit},
  list = ParallelTable[
    {Nc, MaxElectrons[53 / (10 * Nc), Ni, reps]}, {Nc, Nc0 - 1, Nc0 + 1, 1 / 2}];
  Print[list];
  listForFit = {#[[1]], #[[2, 1]]} & /@ list;
  errorslist = #[[2, 2]] & /@ list;
  fit = NonlinearModelFit[listForFit,
    p1 Exp[p2 x] + p3, {p1, p2, p3}, x, Weights → 1 / errorslist^2];
  f = (Log[(ϒ - p3) / p1] / p2);
  error = {D[f, p1], D[f, p2], D[f, p3]}.fit["CovarianceMatrix"].
     Transpose[{D[f, p1], D[f, p2], D[f, p3]}] /. fit["BestFitParameters"];
  Around[f, Sqrt[error]] /. fit["BestFitParameters"]
 ]
```

In[ ]:=
```
FindBreakdownNCSC[40, 50, 34 / 10, 500]
```

(kernel 4) 0.63622

(kernel 3) 0.645566

(kernel 4) −0.746137

(kernel 2) −0.756372

(kernel 1) −0.665681

$$\left\{\left\{\frac{12}{5}, 17.6 \pm 1.3\right\}, \left\{\frac{29}{10}, 31.3 \pm 2.1\right\}, \left\{\frac{17}{5}, 47.4 \pm 3.4\right\}, \left\{\frac{39}{10}, 83. \pm 6.\right\}, \left\{\frac{22}{5}, 101. \pm 7.\right\}\right\}$$

Out[ ]= 3.41 ± 0.07

NC = 10

In[•]:= **FindBreakdownNCSC[30, 50, 4, 500]**

(kernel 4) 0.659182

(kernel 3) 0.754227

(kernel 4) −0.645615

(kernel 2) 0.691999

(kernel 1) −0.708711

During evaluation of In[1047]:=

$$\left\{\{3,\ 26.9\pm1.9\},\ \left\{\frac{7}{2},\ 41.9\pm2.7\right\},\ \{4,\ 74.\pm5.\},\ \left\{\frac{9}{2},\ 92.\pm6.\right\},\ \{5,\ 128.\pm9.\}\right\}$$

Out[•]= 3.63 ± 0.07

In[•]:= **FindBreakdownNCSC[18, 50, 4, 500]**

(kernel 4) −0.858654

(kernel 3) −0.599666

(kernel 4) 0.731309

(kernel 2) −0.75725

(kernel 1) −0.616007

During evaluation of In[1043]:=

$$\left\{\{3,\ 17.8\pm1.4\},\ \left\{\frac{7}{2},\ 30.2\pm2.1\right\},\ \{4,\ 31.0\pm2.1\},\ \left\{\frac{9}{2},\ 52.\pm4.\right\},\ \{5,\ 60.\pm4.\}\right\}$$

Out[•]= 4.62 ± 0.18

In[•]:= **FindBreakdownNISC[53 / 100, 50, 11.1, 1000]**

(kernel 4) −1.

(kernel 3) 0.367845

(kernel 2) -6.29888×10^{-9}

(kernel 1) −1.

(kernel 4) 0.938665

(kernel 3) 0.22094

(kernel 2) 0.936544

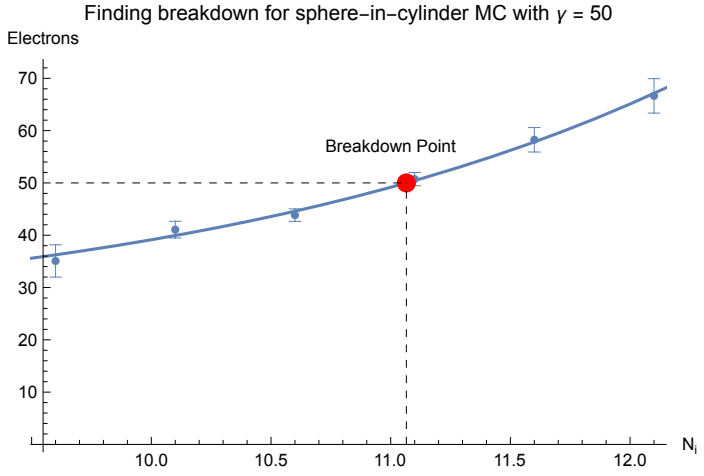
During evaluation of In[978]:=

{{9.6, 35.1 ± 3.1}, {10.1, 41.1 ± 1.6}, {10.6, 43.8 ± 1.2},
{11.1, 50.7 ± 1.3}, {11.6, 58.2 ± 2.3}, {12.1, 66.6 ± 3.3}, {12.6, 99. ± 10.}}

Out[•]= 11.09 ± 0.07

```
In[•]:= electronvals = {{9.6`, 35.1 ±3.1}, {10.1`, 41.1 ±1.6},
          {10.6`, 43.8 ±1.2}, {11.1`, 50.7 ±1.3}, {11.6`, 58.2 ±2.3}, {12.1`, 66.6 ±3.3}}
        listForFit = {#[[1]], #[[2, 1]]} & /@ electronvals;
        errorslist = #[[2, 2]] & /@ electronvals;
        fit = NonlinearModelFit[listForFit,
            p1 Exp[p2 x] + p3, {p1, p2, p3}, x, Weights → 1 / errorslist^2];
        f = (Log[(50 - p3) / p1] / p2);
        error = {D[f, p1], D[f, p2], D[f, p3]}.fit["CovarianceMatrix"].
             Transpose[{D[f, p1], D[f, p2], D[f, p3]}] /. fit["BestFitParameters"];
        res = Around[f, Sqrt[error]] /. fit["BestFitParameters"]

Out[•]= {{9.6, 35.1 ±3.1}, {10.1, 41.1 ±1.6}, {10.6, 43.8 ±1.2},
         {11.1, 50.7 ±1.3}, {11.6, 58.2 ±2.3}, {12.1, 66.6 ±3.3}}

Out[•]= 11.07 ±0.04

In[•]:= res[[1]]

Out[•]= 11.0654

In[•]:= fig7 = Show[ListPlot[electronvals], Plot[fit[x], {x, 9, 16}],
          Graphics[{Red, PointSize[0.03], Point[{res[[1]], 50}]}], Graphics[{Dashed,
            Black, Line[{{9.6, 50}, {res[[1]], 50}}], Line[{{res[[1]], 0}, {res[[1]], 50}}]}],
          Graphics[Text["Breakdown Point", {11, 57}]],
          AxesLabel → {Subscript["N", "i"], "Electrons"},
          PlotLabel → "Finding breakdown for sphere-in-cylinder MC with ɣ = 50"]
```

Out[•]=




```
In[•]:= Export["/Users/wbowers/Documents/stem-fellowship/figures/scanning-ni.TIFF",
         fig7, ImageResolution → 600]

Out[•]= /Users/wbowers/Documents/stem-fellowship/figures/scanning-ni.TIFF

In[•]:= FindBreakdownNISC[53 / 178, 50, 11.1, 500]
```

(kernel 4) -4.36402×10^{-9}

(kernel 3) -1.28008×10^{-8}

(kernel 2) $-1.$

(kernel 4) -7.82048×10^{-9}

(kernel 1) 2.20918×10^{-33}

(kernel 1) General::munfl : $2.20918 \times 10^{-33\,10}$ is too small to represent as a normalized machine number; precision may be lost.

(kernel 1) General::munfl : $2.20918 \times 10^{-33\,10}$ is too small to represent as a normalized machine number; precision may be lost.

(kernel 1) General::munfl : $2.20918 \times 10^{-33\,10}$ is too small to represent as a normalized machine number; precision may be lost.

(kernel 1) General::stop : Further output of General::munfl will be suppressed during this calculation.

(kernel 3) $1.$

(kernel 2) -1.30523×10^{-8}

{{9.6, 26.8 ±0.8}, {10.1, 34.9 ±1.1}, {10.6, 42.1 ±1.4}, {11.1, 55. ±6.}, {11.6, 75. ±8.}, {12.1, 73.2 ±2.9}, {12.6, 95. ±4.}}

Out[•]= 10.99 ± 0.07

In[•]:= **FindBreakdownNISC[53 / 200, 50, 13, 100]**

Out[•]= $Aborted

(kernel 4) -0.934342

(kernel 3) -5.14173×10^{-9}

(kernel 2) $-1.$

(kernel 4) $1.$

(kernel 1) $0.$

(kernel 3) -0.912496

(kernel 2) $-1.$

During evaluation of In[982]:=

$$\left\{\left\{\frac{23}{2}, 55. \pm 5.\right\}, \{12, 65. \pm 11.\}, \left\{\frac{25}{2}, 101. \pm 9.\right\},\right.$$
$$\left.\{13, 103. \pm 12.\}, \left\{\frac{27}{2}, 154. \pm 28.\right\}, \{14, 183. \pm 67.\}, \left\{\frac{29}{2}, 180. \pm 15.\right\}\right\}$$

Out[•]= 11.38 ± 0.14

In[•]:= **FindBreakdownNISC[53 / 400, 50, 13, 200]**

(kernel 4) $-1.$

(kernel 3) -4.96763×10^{-9}

(kernel 2) $1.$

(kernel 4) -0.26817

(kernel 1) -4.1359×10^{-24}

(kernel 3) -6.18105×10^{-9}

(kernel 2) $1.$

During evaluation of In[989]:=

$$\left\{\left\{\frac{23}{2}, 18.7 \pm 2.6\right\}, \{12, 23.5 \pm 0.8\}, \left\{\frac{25}{2}, 29.4 \pm 1.3\right\},\right.$$
$$\left.\{13, 38.4 \pm 2.0\}, \left\{\frac{27}{2}, 46. \pm 5.\right\}, \{14, 59. \pm 7.\}, \left\{\frac{29}{2}, 69.0 \pm 3.2\right\}\right\}$$

Out[ ]= 13.680 ± 0.029

In[ ]:= **FindBreakdownNISC[53 / 700, 50, 17, 100]**

(kernel 4) 0.589819

(kernel 3) $-1.$

(kernel 4) 0.514641

(kernel 2) 0.214126

(kernel 3) -0.907422

(kernel 1) -0.913193

(kernel 2) $1.$

During evaluation of In[986]:=

$$\left\{\left\{\frac{31}{2}, 28.9 \pm 2.3\right\}, \{16, 30.6 \pm 2.2\}, \left\{\frac{33}{2}, 45. \pm 12.\right\},\right.$$
$$\left.\{17, 46. \pm 4.\}, \left\{\frac{35}{2}, 60.9 \pm 2.8\right\}, \{18, 72. \pm 14.\}, \left\{\frac{37}{2}, 96. \pm 7.\right\}\right\}$$

Out[ ]= 17.12 ± 0.06

In[ ]:= **FindBreakdownNISC[53 / 1100, 50, 23, 100]**

(kernel 4) $-1.$

(kernel 3) -0.970182

(kernel 4) -0.306198

(kernel 2) -0.918607

(kernel 3) -0.265945

(kernel 1) -1.03398×10^{-25}

(kernel 2) -4.71216×10^{-9}

$$\left\{\left\{\frac{43}{2}, 56. \pm 15.\right\}, \{22, 60. \pm 4.\}, \left\{\frac{45}{2}, 87. \pm 7.\right\},\right.$$
$$\left.\{23, 100. \pm 6.\}, \left\{\frac{47}{2}, 124. \pm 11.\right\}, \{24, 163. \pm 9.\}, \left\{\frac{49}{2}, 214. \pm 16.\right\}\right\}$$

Out[•]= 21.53 ± 0.19

In[•]:= **FindBreakdownNISC[53 / 1500, 50, 24, 80]**

(kernel 4) 2.06795×10^{-25}

(kernel 3) -6.58048×10^{-9}

(kernel 4) -0.8947

(kernel 2) 0.305773

(kernel 3) -0.355367

(kernel 1) -0.882098

(kernel 2) 0.488683

$$\left\{\left\{\frac{45}{2}, 24.3 \pm 2.9\right\}, \{23, 35. \pm 4.\}, \left\{\frac{47}{2}, 32.8 \pm 3.2\right\},\right.$$
$$\left.\{24, 41.1 \pm 3.1\}, \left\{\frac{49}{2}, 48. \pm 5.\right\}, \{25, 54. \pm 5.\}, \left\{\frac{51}{2}, 61. \pm 7.\right\}\right\}$$

Out[•]= 24.71 ± 0.12

```
In[ ]:= scbreakdown = {{5.3 / 3.41 ± 0.07, 40}, {5.3 / 3.63 ± 0.07, 30}, {5.3 / 4.62 ± 0.18, 18},
          {53 / 100, 11.09 ± 0.07}, {53 / 200, 11.38 ± 0.14}, {53 / 400, 13.680 ± 0.029},
          {53 / 700, 17.12 ± 0.06}, {53 / 1100, 21.53 ± 0.19}, {53 / 1500, 24.71 ± 0.12}}
        sbreakdownscaled = {1 / #[[1]], 15 * #[[2]]} & /@ scbreakdown
```

$$\left\{\{1.557 \pm 0.031, 40\}, \{1.459 \pm 0.028, 30\}, \{1.15 \pm 0.04, 18\}, \left\{\frac{53}{100}, 11.09 \pm 0.07\right\}, \left\{\frac{53}{200}, 11.38 \pm 0.14\right\}, \left\{\frac{53}{400}, 13.680 \pm 0.029\right\}, \left\{\frac{53}{700}, 17.12 \pm 0.06\right\}, \left\{\frac{53}{1100}, 21.53 \pm 0.19\right\}, \left\{\frac{53}{1500}, 24.71 \pm 0.12\right\}\right\}$$

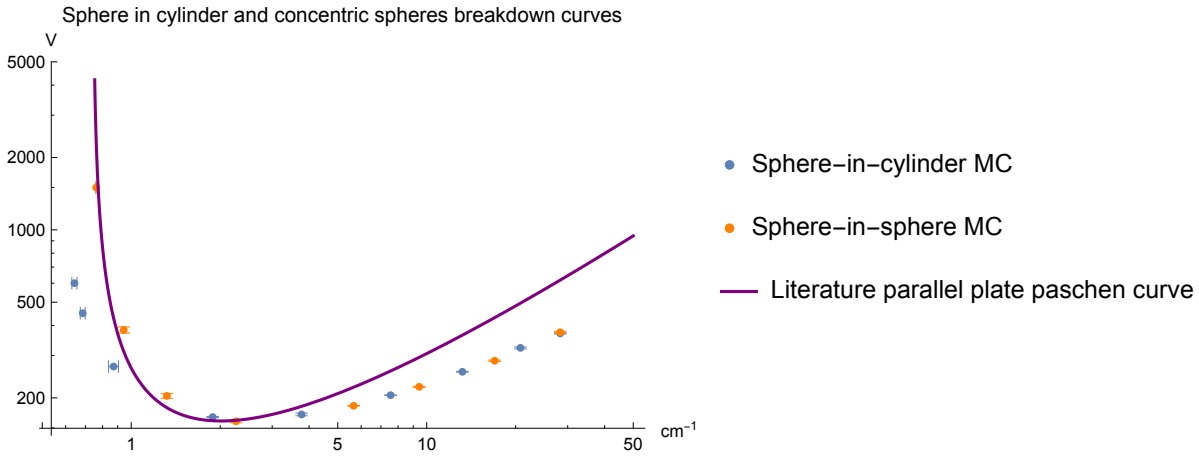
$$\left\{\{0.642 \pm 0.013, 600\}, \{0.685 \pm 0.013, 450\}, \{0.872 \pm 0.034, 270\}, \left\{\frac{100}{53}, 166.4 \pm 1.0\right\}, \left\{\frac{200}{53}, 170.6 \pm 2.2\right\}, \left\{\frac{400}{53}, 205.2 \pm 0.4\right\}, \left\{\frac{700}{53}, 256.8 \pm 1.0\right\}, \left\{\frac{1100}{53}, 322.9 \pm 2.8\right\}, \left\{\frac{1500}{53}, 370.7 \pm 1.7\right\}\right\}$$

```
In[ ]:= ssbreakdownscaled = {#[[1]] / 5.3, 15 * #[[2]]} & /@ ssbd
```

```
Out[ ]= {{0.7616 ± 0.0035, 1500}, {0.943396, 383. ± 11.}, {1.32075, 204. ± 5.},
         {2.26415, 159.8 ± 2.2}, {5.66038, 185.4 ± 0.5}, {9.43396, 222.4 ± 1.4},
         {16.9811, 285.4 ± 1.8}, {28.3019, 375.0 ± 3.1}}
```

```
In[ ]:= fig12 =
         Show[ListLogLogPlot[sbreakdownscaled, PlotLegends → {"Sphere-in-cylinder MC"}],
          ListLogLogPlot[ssbreakdownscaled, PlotStyle → Orange,
           PlotLegends → {"Sphere-in-sphere MC"}], LogLogPlot[
           15 (5.3) Nc / (Log[Nc * 5.3] - Log[Log[51]]), {Nc, 0, 50}, PlotStyle → Purple,
           PlotLegends → {"Literature parallel plate paschen curve"}], PlotRange → All,
          PlotLabel → "Sphere in cylinder and concentric spheres breakdown curves",
          AxesLabel → {Superscript["cm", "-1"], "V"}]
```

Out[ ]=



In[ ]:=
```
Export[
 "/Users/wbowers/Documents/stem-fellowship/tif-figs/ss-and-sc-breakdown.TIFF",
 fig12, ImageResolution → 600]
```

Out[ ]=
```
/Users/wbowers/Documents/stem-fellowship/tif-figs/ss-and-sc-breakdown.TIFF
```

Plotting fusor data

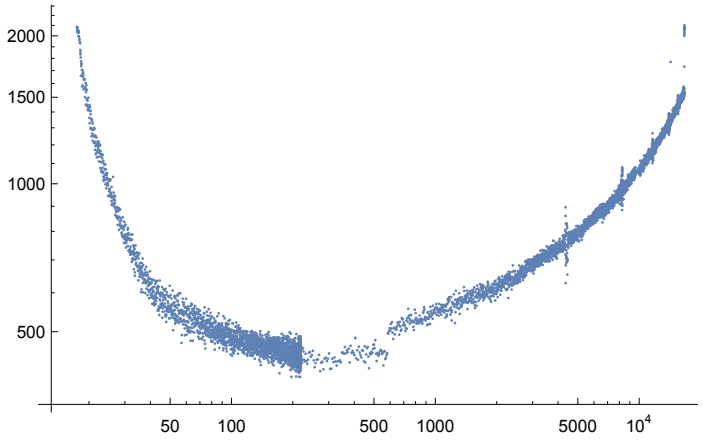
In[ ]:=
```
data = Drop[Import["/Users/wbowers/Downloads/RAWpashendata.txt", "Table"], 1]
```

Out[ ]=
```
{{17.4373, 2086.77}, {17.4627, 2052.92}, {17.4878, 2069.5},
 {17.5153, 2036.34}, {17.549, 2081.93}, {17.5836, 2041.17}, {17.6141, 2037.03},
 {17.641, 2048.08}, {17.6684, 2031.5}, {17.7051, 2051.54}, {17.7484, 1998.34},
 {17.7916, 2043.25}, {17.8291, 2030.12}, {17.8664, 1991.44},
 {17.9028, 1962.42}, {17.9676, 1967.26}, ··· 7141 ···, {16656.5, 2019.76},
 {16660, 2057.06}, {16660, 2035.65}, {16660, 2054.99}, {16660, 2073.64},
 {16660, 2073.64}, {16660, 2073.64}, {16663.4, 2073.64}, {16666.9, 2081.24},
 {16670.7, 2081.24}, {16677.3, 2079.17}, {16684, 2102.66}, {16690.9, 2073.64},
 {16697.9, 2077.79}, {16704.9, 2082.62}, {----------, --------}}
```

large output | show less | show more | show all | set si | ze limit...

In[ ]:=
```
ListLogLogPlot[data]
```

Out[ ]=



In[ ]:=
```
b = 365
```

Out[ ]=
```
365
```

In[ ]:=
```
a = 15
```

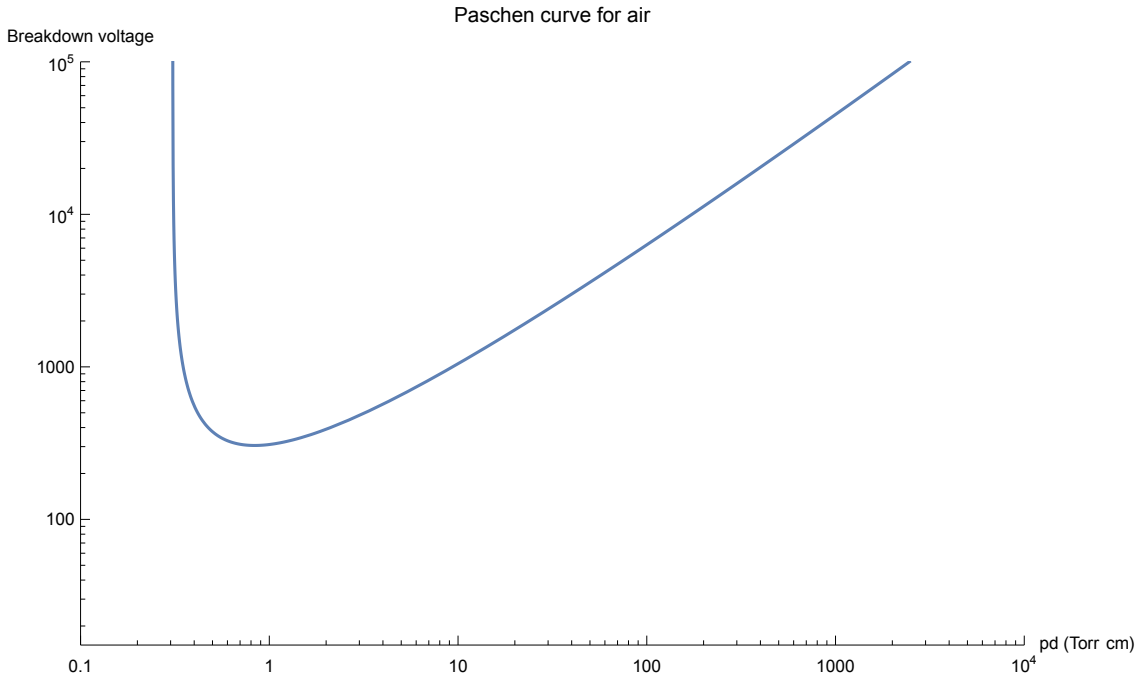
Out[ ]=
```
15
```

In[ ]:=
```
k = Log[a / Log[1 + 1 / (0.01)]]
```

Out[ ]=
```
1.17871
```

In[ ]:=
```
fig1 = LogLogPlot[b x / (Log[x] + k),
  {x, 0.1, 10 000}, PlotRange → {{0.1, 10 000}, {15, 100 000}},
  AxesOrigin → {0.1, 15}, PlotLabel → "Paschen curve for air",
  AxesLabel → {"pd (Torr cm)" , "Breakdown voltage"}]
```

Out[ ]=

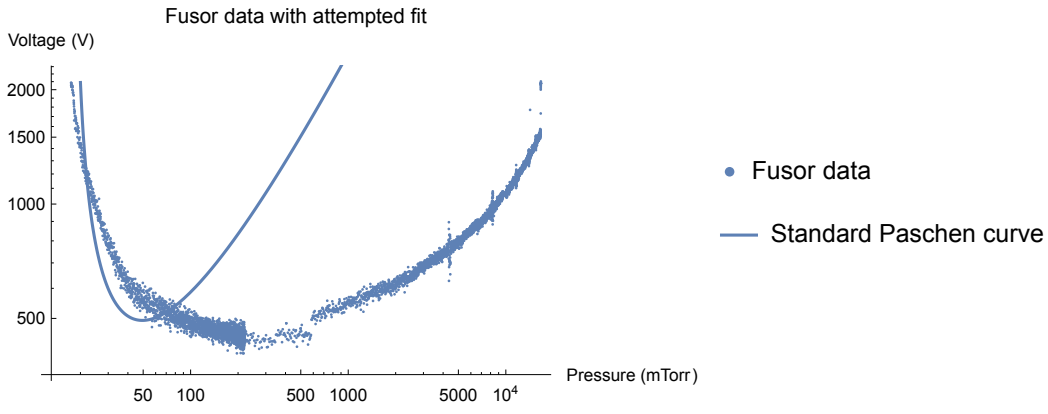


In[ ]:=
```
Export["/Users/wbowers/Documents/stem-fellowship/tif-figs/air-paschen-curve.TIFF",
 fig1, ImageResolution → 600]
```

Out[ ]=
```
/Users/wbowers/Documents/stem-fellowship/tif-figs/air-paschen-curve.TIFF
```

In[ ]:=
```
fig2 = Show[ListLogLogPlot[data, PlotLegends → {"Fusor data"},
    PlotStyle → {PointSize[0.004]}], LogLogPlot[10 x / (Log[x] - 2.9),
    {x, 20, 1000}, PlotLegends → {"Standard Paschen curve"}],
  AxesLabel → {"Pressure (mTorr)", "Voltage (V)"},
  PlotLabel → "Fusor data with attempted fit"]
```

Out[ ]=



In[ ]:=
```
Export["/Users/wbowers/Documents/stem-fellowship/tif-figs/attempted-fit.TIFF",
 fig2, ImageResolution → 600]
```

Out[ ]=
```
/Users/wbowers/Documents/stem-fellowship/tif-figs/attempted-fit.TIFF
```