In[1]:=

```
(*Set initial directory*)
directory = NotebookDirectory[];



(*Display menu for nuclear species selection*)


Print["Select Nuclear Species:"];
Print["1. U-235_Data_JENDL5_CalculationData"];
Print["2. U-233_Data_JENDL5_CalculationData"];
Print["3. U-238_Data_JENDL5_CalculationData"];
Print["4. Th-232_Data_JENDL5_CalculationData"];
Print["5. Np-237_Data_JENDL5_CalculationData"];
Print["6. Pu-239_Data_JENDL5_CalculationData"];
Print["7. Pu-240_Data_JENDL5_CalculationData"];
Print["8. Pu-242_Data_JENDL5_CalculationData"];
Print["9. Am-241_Data_JENDL5_CalculationData"];

(*Get user selection*)
choice = Input["Enter number (1-9): "];

(*Load file based on selection*)
fileName = Switch[choice, 1, "U-235_Data_JENDL5_CalculationData.m", 2,
    "U-233_Data_JENDL5_CalculationData.m", 3, "U-238_Data_JENDL5_CalculationData.m", 4,
    "Th-232_Data_JENDL5_CalculationData.m", 5, "Np-237_Data_JENDL5_CalculationData.m",
    6, "Pu-239_Data_JENDL5_CalculationData.m", 7,
    "Pu-240_Data_JENDL5_CalculationData.m", 8, "Pu-242_Data_JENDL5_CalculationData.m",
    9, "Am-241_Data_JENDL5_CalculationData.m", _, (Print["Invalid selection"];
     Abort[])];



(*Load file using full path*)
fullPath = FileNameJoin[{directory, fileName}];
Print["Attempting to load file from: ", fullPath];

If[FileExistsQ[fullPath], Get[fullPath];
  Print["Successfully loaded: ", fileName],
  Print["Error: File not found at ", fullPath]];

(*2*)


(*Initial Calculations for Nuclear Parameters*)
fragmentZ1 = atomicNumber / 2; (*First fragment atomic number*)
fragmentZ2 = atomicNumber / 2; (*Second fragment atomic number*)
reducedMass = N[Sqrt[fragmentZ1 * fragmentZ2 / (fragmentZ1 + fragmentZ2)]];


fitStartZ = 28; (*Starting atomic number for fitting*)
fitEndZ = atomicNumber - fitStartZ; (*Ending atomic number for fitting*)
```

```
fitStartIndex = fitStartZ - 22;(*Starting index for fitting data*)
fitEndIndex = fitEndZ - 22;(*Ending index for fitting data*)
(*Initialize distance parameter*)



(*Incident Neutron Kinetic Energy*)
neutronEnergy1 = 0;(*0.0253 eV*)
neutronEnergy2 = 0.5;(*500 keV*)
neutronEnergy3 = 14;(*14 MeV*)



effectiveDistance = .;

(*Initialize energy-dependent distance parameters*)
effectiveDistance0253eV = .;
effectiveDistance500keV = .;
effectiveDistance14MeV = .;

(*Initialize variable lists for each energy region*)
distanceVars0253eV = {};
distanceVars500keV = {};
distanceVars14MeV = {};

(*Initialize fermi correction lists*)
fermiVars0253eV = {};
fermiVars500keV = {};
fermiVars14MeV = {};

(*Initialize fermi correction energies*)
fermiEnergy1 = .;
fermiEnergy2 = .;
fermiEnergy3 = .;

(*Initialize various parameter lists*)
paramList0253eV = {};
paramList500keV = {};
paramList14MeV = {};
paramList5 = {};
paramList6 = {};
paramList7 = {};

fermiEnergies = {};
variablesList = {};
fissionYields = {};
neutronVars = {};
effectiveDistances = {};
distanceParams = {};

paramList50 = {};
paramList60 = {};
paramList70 = {};
```

```
Clear[theoreticalYield0253eV, theoreticalYield500keV, theoreticalYield14MeV,
 fittedYield0253eV, fittedYield500keV, fittedYield14MeV, optResult0253eV,
 optResult500keV, optResult14MeV, fermiVars0253eV, fermiVars500keV, fermiVars14MeV]

(*Mass retrieval function definition*)
getNuclearMass[Z_, A_] :=
 Module[{elementName, isotopeName, mass, numericMass}, (*Validate atomic number*)
  If[! NumberQ[Z] || Z < 1 || Z > 118, Return[Missing["NotAvailable"]]];
  (*Get element name*)elementName = ElementData[Z, "Name"];
  If[elementName === $Failed, Return[Missing["NotAvailable"]]];
  (*Format element name*)elementName =
   StringReplacePart[elementName, ToUpperCase[StringTake[elementName, 1]], {1, 1}];
  (*Create isotope name*)isotopeName = elementName <> ToString[A];
  (*Get mass data*)mass = IsotopeData[isotopeName, "AtomicMass"];
  (*Return numeric mass or missing value*)If[mass === Missing["NotAvailable"],
   Missing["NotAvailable"], numericMass = QuantityMagnitude[mass];
   numericMass]]


(*Energy pattern selection*)
{startEnergyIndex, endEnergyIndex} =
  Switch[energyPattern, 1, {1, 1}, (*0.0253 eV only*)2, {1, 2},
   (*0.0253 eV and 500 keV*)3, {1, 3}, (*All three energies*)4,
   {2, 2}, (*500 keV only*)5, {2, 3}  (*500 keV and 14 MeV*)];

(*Isotope existence check function*)
isIsotopeStable[z_, n_] :=
 Module[{element, isotope, mass}, element = ElementData[z, "Name"];
  element = StringReplacePart[element, ToUpperCase[StringTake[element, 1]], {1, 1}];
  isotope = element <> ToString[z + n];
  mass = IsotopeData[isotope, "AtomicMass"];
  mass =!= Missing["NotAvailable"] && NumberQ[QuantityMagnitude[mass]]]


(*3*)(*Main Nuclear Fission Calculation Function*)
CalculateFissionYields[energyPattern_] := Module[{dataTemp, results = {}},
  For[energyIndex = startEnergyIndex, energyIndex ≤ endEnergyIndex, energyIndex++,
   (*Set parameters for each energy region*){promptNeutronCount, incidentEnergy,
     effectiveDistance, variableList, yieldList, neutronVariables} =
    Switch[energyIndex, 1, {promptNeutrons1, neutronEnergy1, effectiveDistance0253eV,
      distanceVars0253eV, fissionYield0253eV, fermiVars0253eV}, (*Thermal*)
     2, {promptNeutrons2, neutronEnergy2, effectiveDistance500keV,
      distanceVars500keV, fissionYield500keV, fermiVars500keV}, (*Intermediate*)
     3, {promptNeutrons3, neutronEnergy3, effectiveDistance14MeV,
      distanceVars14MeV, fissionYield14MeV, fermiVars14MeV}   (*Fast*)];
   dataTemp = Reap[For[protonNumber1 = 23, protonNumber1 ≤ atomicNumber - 23,
       protonNumber1++, For[neutronCount1 = 0, neutronCount1 ≤ neutronNumber,
        neutronCount1++, protonNumber2 = atomicNumber - protonNumber1;
        neutronCount2 = neutronNumber - neutronCount1 - Round[promptNeutronCount];
        massNumber1 = protonNumber1 + neutronCount1;
        massNumber2 = protonNumber2 + neutronCount2;


        If[(protonNumber1 == 22 && 16 <= neutronCount1 <= 41) ||
```

```
(protonNumber1 == 23 && 17 <= neutronCount1 <= 42) ||
(protonNumber1 == 24 && 18 <= neutronCount1 <= 43) ||
(protonNumber1 == 25 && 19 <= neutronCount1 <= 44) ||
(protonNumber1 == 26 && 19 <= neutronCount1 <= 46) ||
(protonNumber1 == 27 && 20 <= neutronCount1 <= 48) ||
(protonNumber1 == 28 && 20 <= neutronCount1 <= 50) ||
(protonNumber1 == 29 && 23 <= neutronCount1 <= 51) ||
(protonNumber1 == 30 && 24 <= neutronCount1 <= 53) ||
(protonNumber1 == 31 && 25 <= neutronCount1 <= 55) ||
(protonNumber1 == 32 && 26 <= neutronCount1 <= 57) ||
(protonNumber1 == 33 && 27 <= neutronCount1 <= 59) ||
(protonNumber1 == 34 && 31 <= neutronCount1 <= 60) ||
(protonNumber1 == 35 && 32 <= neutronCount1 <= 62) ||
(protonNumber1 == 36 && 33 <= neutronCount1 <= 64) ||
(protonNumber1 == 37 && 34 <= neutronCount1 <= 65) ||
(protonNumber1 == 38 && 35 <= neutronCount1 <= 67) ||
(protonNumber1 == 39 && 37 <= neutronCount1 <= 69) ||
(protonNumber1 == 40 && 38 <= neutronCount1 <= 70) ||
(protonNumber1 == 41 && 40 <= neutronCount1 <= 72) ||
(protonNumber1 == 42 && 41 <= neutronCount1 <= 73) ||
(protonNumber1 == 43 && 42 <= neutronCount1 <= 75) ||
(protonNumber1 == 44 && 43 <= neutronCount1 <= 76) ||
(protonNumber1 == 45 && 44 <= neutronCount1 <= 77) ||
(protonNumber1 == 46 && 45 <= neutronCount1 <= 78) ||
(protonNumber1 == 47 && 46 <= neutronCount1 <= 83) ||
(protonNumber1 == 48 && 47 <= neutronCount1 <= 84) ||
(protonNumber1 == 49 && 48 <= neutronCount1 <= 86) ||
(protonNumber1 == 50 && 49 <= neutronCount1 <= 87) ||
(protonNumber1 == 51 && 52 <= neutronCount1 <= 88) ||
(protonNumber1 == 52 && 53 <= neutronCount1 <= 90) ||
(protonNumber1 == 53 && 55 <= neutronCount1 <= 91) ||
(protonNumber1 == 54 && 56 <= neutronCount1 <= 93) ||
(protonNumber1 == 55 && 57 <= neutronCount1 <= 96) ||
(protonNumber1 == 56 && 58 <= neutronCount1 <= 97) ||
(protonNumber1 == 57 && 60 <= neutronCount1 <= 98) ||
(protonNumber1 == 58 && 61 <= neutronCount1 <= 99) ||
(protonNumber1 == 59 && 62 <= neutronCount1 <= 100) ||
(protonNumber1 == 60 && 64 <= neutronCount1 <= 101) ||
(protonNumber1 == 61 && 65 <= neutronCount1 <= 102) ||
(protonNumber1 == 62 && 66 <= neutronCount1 <= 103) ||
(protonNumber1 == 63 && 67 <= neutronCount1 <= 104) ||
(protonNumber1 == 64 && 70 <= neutronCount1 <= 105) ||
(protonNumber1 == 65 && 71 <= neutronCount1 <= 106) ||
(protonNumber1 == 66 && 72 <= neutronCount1 <= 107) ||
(protonNumber1 == 67 && 73 <= neutronCount1 <= 108) ||
(protonNumber1 == 68 && 75 <= neutronCount1 <= 109) ||
(protonNumber1 == 69 && 76 <= neutronCount1 <= 110) ||
(protonNumber1 == 70 && 78 <= neutronCount1 <= 111) ||
(protonNumber1 == 71 && 79 <= neutronCount1 <= 111) ||
(protonNumber1 == 72 && 81 ≤ neutronCount1 ≤ 116) ||
```

```
  (protonNumber1 == 73 && 82 ≤ neutronCount1 ≤ 117) ||
  (protonNumber1 == 74 && 84 ≤ neutronCount1 ≤ 118) ||
  (protonNumber1 == 75 && 85 ≤ neutronCount1 ≤ 119),
 (*Check nuclear chart range for second fragment*)
 If[(protonNumber2 == 22 && 16 <= neutronCount2 <= 41) ||
   (protonNumber2 == 23 && 17 <= neutronCount2 <= 42) ||
   (protonNumber2 == 24 && 18 <= neutronCount2 <= 43) ||
   (protonNumber2 == 25 && 19 <= neutronCount2 <= 44) ||
   (protonNumber2 == 26 && 19 <= neutronCount2 <= 46) ||
   (protonNumber2 == 27 && 20 <= neutronCount2 <= 48) ||
   (protonNumber2 == 28 && 20 <= neutronCount2 <= 50) ||
   (protonNumber2 == 29 && 23 <= neutronCount2 <= 51) ||
   (protonNumber2 == 30 && 24 <= neutronCount2 <= 53) ||
   (protonNumber2 == 31 && 25 <= neutronCount2 <= 55) ||
   (protonNumber2 == 32 && 26 <= neutronCount2 <= 57) ||
   (protonNumber2 == 33 && 27 <= neutronCount2 <= 59) ||
   (protonNumber2 == 34 && 31 <= neutronCount2 <= 60) ||
   (protonNumber2 == 35 && 32 <= neutronCount2 <= 62) ||
   (protonNumber2 == 36 && 33 <= neutronCount2 <= 64) ||
   (protonNumber2 == 37 && 34 <= neutronCount2 <= 65) ||
   (protonNumber2 == 38 && 35 <= neutronCount2 <= 67) ||
   (protonNumber2 == 39 && 37 <= neutronCount2 <= 69) ||
   (protonNumber2 == 40 && 38 <= neutronCount2 <= 70) ||
   (protonNumber2 == 41 && 40 <= neutronCount2 <= 72) ||
   (protonNumber2 == 42 && 41 <= neutronCount2 <= 73) ||
   (protonNumber2 == 43 && 42 <= neutronCount2 <= 75) ||
   (protonNumber2 == 44 && 43 <= neutronCount2 <= 76) ||
   (protonNumber2 == 45 && 44 <= neutronCount2 <= 77) ||
   (protonNumber2 == 46 && 45 <= neutronCount2 <= 78) ||
   (protonNumber2 == 47 && 46 <= neutronCount2 <= 83) ||
   (protonNumber2 == 48 && 47 <= neutronCount2 <= 84) ||
   (protonNumber2 == 49 && 48 <= neutronCount2 <= 86) ||
   (protonNumber2 == 50 && 49 <= neutronCount2 <= 87) ||
   (protonNumber2 == 51 && 52 <= neutronCount2 <= 88) ||
   (protonNumber2 == 52 && 53 <= neutronCount2 <= 90) ||
   (protonNumber2 == 53 && 55 <= neutronCount2 <= 91) ||
   (protonNumber2 == 54 && 56 <= neutronCount2 <= 93) ||
   (protonNumber2 == 55 && 57 <= neutronCount2 <= 96) ||
   (protonNumber2 == 56 && 58 <= neutronCount2 <= 97) ||
   (protonNumber2 == 57 && 60 <= neutronCount2 <= 98) ||
   (protonNumber2 == 58 && 61 <= neutronCount2 <= 99) ||
   (protonNumber2 == 59 && 62 <= neutronCount2 <= 100) ||
   (protonNumber2 == 60 && 64 <= neutronCount2 <= 101) ||
   (protonNumber2 == 61 && 65 <= neutronCount2 <= 102) ||
   (protonNumber2 == 62 && 66 <= neutronCount2 <= 103) ||
   (protonNumber2 == 63 && 67 <= neutronCount2 <= 104) ||
   (protonNumber2 == 64 && 70 <= neutronCount2 <= 105) ||
   (protonNumber2 == 65 && 71 <= neutronCount2 <= 106) ||
   (protonNumber2 == 66 && 72 <= neutronCount2 <= 107) ||
   (protonNumber2 == 67 && 73 <= neutronCount2 <= 108) ||
```

```
      (protonNumber2 == 68 && 75 <= neutronCount2 <= 109) ||
      (protonNumber2 == 69 && 76 <= neutronCount2 <= 110) ||
      (protonNumber2 == 70 && 78 <= neutronCount2 <= 111) ||
      (protonNumber2 == 71 && 79 <= neutronCount2 <= 111) ||
      (protonNumber2 == 72 && 81 ≤ neutronCount2 ≤ 116) ||
      (protonNumber2 == 73 && 82 ≤ neutronCount2 ≤ 117) ||
      (protonNumber2 == 74 && 84 ≤ neutronCount2 ≤ 118) ||
      (protonNumber2 == 75 && 85 ≤ neutronCount2 ≤ 119),
     (*Calculate proton number ratio*)chargeRatio =
      N[(protonNumber1 / massNumber1) / (protonNumber2 / massNumber2)];



     (*Update variable lists based on energy region*)
     Switch[energyIndex, 1, fermiVars0253eV = Union[AppendTo[distanceVars0253eV,
         effectiveDistance0253eV[protonNumber1, protonNumber2]]],
      2, fermiVars500keV = Union[AppendTo[distanceVars500keV,
         effectiveDistance500keV[protonNumber1, protonNumber2]]],
      3, fermiVars14MeV = Union[AppendTo[distanceVars14MeV,
         effectiveDistance14MeV[protonNumber1, protonNumber2]]]];

     (*Calculate physical parameters*)
     effectiveDistanceVal = effectiveDistance[protonNumber1, protonNumber2];
     coulombEnergy = (1.44 * protonNumber1 * protonNumber2) / effectiveDistanceVal;

     (*Calculating Q value*)
     qValue = (getNuclearMass[atomicNumber, atomicNumber + neutronNumber] -
         getNuclearMass[protonNumber1, massNumber1] -
         getNuclearMass[protonNumber2, massNumber2] -
         promptNeutronCount * 1.008665) * 931.4940954;

     effectiveEnergy = coulombEnergy - qValue;

     (*Calculate fission probability*)
     probability = 1 / (1 + Exp[2 * Pi / (neutronSeparationEnergy + incidentEnergy) *
           Sqrt[protonNumber1 * protonNumber2 / (protonNumber1 + protonNumber2)] /
            reducedMass * effectiveEnergy]);


     Sow[{protonNumber1, probability}, yieldList];]]],
   {yieldList}, Rule][[2, All, 1]];
  (*Process results*)
  fragmentData = Part[yieldList /. dataTemp];
  processYields[data_] :=
   (Total@# / {Length@#, Total@data[[All, 2]] / 2} &) /@ GatherBy[data, First];
  AppendTo[results, {energyIndex, processYields[fragmentData]}];];
 results]

(*Execute main calculation*)
fissionResults = CalculateFissionYields[energyPattern];

(*Process results based on energy pattern*)
{yieldData0253eVCalc, yieldData500keVCalc, yieldData14MeVCalc} =
```

```
  Switch[energyPattern, 1, {fissionResults[[1, 2]], Null, Null},
   2, {fissionResults[[1, 2]], fissionResults[[2, 2]], Null}, 3,
   {fissionResults[[1, 2]], fissionResults[[2, 2]], fissionResults[[3, 2]]},
   4, {Null, fissionResults[[1, 2]], Null}, 5,
   {Null, fissionResults[[1, 2]], fissionResults[[2, 2]]}];
```

```
(*4*)(*Optimization and Result Display Program for Fission Parameters*)
For[energyRegion = startEnergyIndex, energyRegion ≤ endEnergyIndex,
  energyRegion++, (*Setup variables and data for each energy region*)
  {neutronVarList, calcYieldData, fitYieldData, experimentalData, energyDescription,
    theoreticalYield, optimizationResult, fittedData} = Switch[energyRegion, 1,
    {paramList0253eV = Union[fermiVars0253eV], yieldData0253eVCalc, fitYield0253eV =
      yieldData0253eVCalc[[fitStartIndex ;; fitEndIndex]], yieldData0253eV[[
      fitStartIndex ;; fitEndIndex]], "1. Incident Neutron Energy: 0.0253eV",
     theoreticalYield0253eV, optResult0253eV, fittedYield0253eV}, 2,
    {paramList500keV = Union[fermiVars500keV], yieldData500keVCalc,
     fitYield500keV = yieldData500keVCalc[[fitStartIndex ;; fitEndIndex]],
     yieldData500keV[[fitStartIndex ;; fitEndIndex]],
     "2. Incident Neutron Energy: 500keV", theoreticalYield500keV,
     optResult500keV, fittedYield500keV}, 3,
    {paramList14MeV = Union[fermiVars14MeV], yieldData14MeVCalc,
     fitYield14MeV = yieldData14MeVCalc[[fitStartIndex ;; fitEndIndex]],
     yieldData14MeV[[fitStartIndex ;; fitEndIndex]],
     "3. Incident Neutron Energy: 14MeV", theoreticalYield14MeV,
     optResult14MeV, fittedYield14MeV}];
  (*Calculate logarithmic difference between theory and experiment*)
  logDifference = (Log@fitYieldData - Log@experimentalData)[[All, 2]];
```

```
  (*Optimize parameters using least squares method*)
  Switch[energyRegion, 1,
   fittedYield0253eV = optResult0253eVRe1[[2, All, All]];
   theoreticalYield0253eV = fitYield0253eV /. optResult0253eVRe1[[2, All, All]], 2,
   fittedYield500keV = optResult500keVRe1[[2, All, All]];
   theoreticalYield500keV = fitYield500keV /. optResult500keVRe1[[2, All, All]], 3,
   fittedYield14MeV = optResult14MeVRe1[[2, All, All]];
   theoreticalYield14MeV = fitYield14MeV /. optResult14MeVRe1[[2, All, All]]];
```

```
  (*Display results*)
  Print[Style[energyDescription <> ": Effective Fission Distance Reff
       derived from experimental charge distribution", 16]];
```

```
  Print["Analysis Results"];
  Print[
   "Calculation results demonstrating that the effective fission distance Reff " <>
    "derived from optimization calculations
      accurately reproduces experimental values " <>
```

```
    "(confirming agreement between JENDL-5 experimental and theoretical values, " <>
    "and validating calculations using Mathematica ver11.2 FindMinimum)"];


  (*Create visualization plot*)
  plotOptions = {Joined → {True, True}, PlotRange → {{15, 80}, {10^(-12), 5}},
    Epilog → Inset[Style[isotopeName, Bold, 20], Scaled@{0.14, 0.9}],
    PlotMarkers → Automatic, PlotStyle →
     {Directive[PointSize[1/100], Red], Directive[PointSize[1/100], Blue]},
    Frame → True, FrameLabel → {"Atomic Number", "Fission Yield (Independent)"},
    LabelStyle → Directive[Black, 19], FrameTicks → Automatic,
    FrameStyle → {Thick, Thick, Thick, Thick},
    PlotLegends → Placed[PointLegend[Automatic, {"JENDL-5", "Theoretical"},
       Joined → {True, True, True}, Joined → {True, True},
       LabelStyle → Directive[Black, 18], LegendFunction → "Frame",
       LegendLayout → "Column", LegendMarkers → Array[{Graphics@Disk[], 10} &, 3]],
      {{0.65, 0.25}, {1, 0.9}}], AspectRatio → 0.8, ImageSize → 450,
    Epilog → Inset[Style[isotopeName, Bold], Scaled@{0.1, 0.92}]};
  Print[ListLogPlot[{experimentalData, theoreticalYield}, Evaluate[plotOptions]]];


  Print[
   "-------------------------------------------------------------------------
     ---"];];



(*5*)
(*Correlation Analysis of Fragment Charge Product and Effective Fission Distance*)
(*Process data for each energy condition*)
Do[With[{condition = Which[i == 1, {energyPattern == 1 || energyPattern == 2 ||
         energyPattern == 3, correlationData0253eV, fittedYield0253eV,
        "1. Analysis for Incident Neutron Energy: 0.0253eV"}, i == 2,
       {energyPattern == 2 || energyPattern == 3 || energyPattern == 4 || energyPattern == 5,
        correlationData500keV, fittedYield500keV,
        "2. Analysis for Incident Neutron Energy: 500keV"}, i == 3,
       {energyPattern == 3 || energyPattern == 5, correlationData14MeV, fittedYield14MeV,
        "3. Analysis for Incident Neutron Energy: 14MeV"}]}, If[condition[[1]],
    Switch[condition[[2]], correlationData0253eV, correlationData0253eV =
      Thread[{condition[[3]][[All, 1, 1]], condition[[3]][[All, 2]]}][[
       fitStartIndex ;; fitEndIndex]], correlationData500keV, correlationData500keV =
      Thread[{condition[[3]][[All, 1, 1]], condition[[3]][[All, 2]]}][[
       fitStartIndex ;; fitEndIndex]], correlationData14MeV, correlationData14MeV =
      Thread[{condition[[3]][[All, 1, 1]], condition[[3]][[All, 2]]}][[
       fitStartIndex ;; fitEndIndex]]]]], {i, 1, 3}];

Print[
  "-------------------------------------------------------------------------
    -"];


(*Result explanation and plotting*)
Module[{plotOptions = {Joined → {True, True, True}, PlotRange → {{15, 85}, {0.8, 1.3}},
     Frame → True, FrameLabel → {"Atomic Number", "Effective Distance (fm)"},
```

```
      LabelStyle → Directive[Black, 19], FrameTicks → Automatic,
      FrameStyle → {Thick, Thick, Thick, Thick}, AspectRatio → 1.1, ImageSize → 350}},
  (*Define fitting function*)analyzeFittingResults[correlationData_, energyLabel_] :=
   Module[{distanceFunction, coeffA, coeffB, coeffC, chargeNumber, normalizedFormula},
    Print[Style[energyLabel, FontSize → 16]];
    (*Fit quadratic function*)distanceFunction =
     Fit[correlationData, {1, x, x^2}, x];
    (*Extract coefficients*){coeffC, coeffB, coeffA} =
     CoefficientList[distanceFunction, x];
    (*Calculate characteristic charge number*)kValue = -coeffA;
    chargeNumber = coeffB / kValue;
    mValue = coeffC;
    (*Display formula*)Print[Column[{Style[HoldForm[Reff] ==
         N[mValue, 6] + N[kValue, 6] * (N[chargeNumber, 6] - x) * x, FontSize → 16]}]];
    distanceFunction (*Return function for later use*)];
  Print["\nAnalysis Results:"];
  Print["1. The effective fission distance
     Reff shows quadratic dependence on fragment charge"];
  Print["2. This dependence reflects fundamental laws of charge
     distribution in fission process"];
  Print["3. Similar dependence is maintained across different incident energies\n"];
  (*Create comparison plot*)plotData = Select[{correlationData0253eV,
     correlationData500keV, correlationData14MeV}, Length[#] > 0 &];

  (*Create data-dependent color and label lists*)colors = {};
labels = {};
If[Length[correlationData0253eV] > 0, AppendTo[colors, Blue];
   AppendTo[labels, "0.0253 eV"]];
If[Length[correlationData500keV] > 0, AppendTo[colors, Green];
   AppendTo[labels, "500 keV"]];
If[Length[correlationData14MeV] > 0, AppendTo[colors, Red];
   AppendTo[labels, "14 MeV"]];

  plotData = Select[{correlationData0253eV,
     correlationData500keV, correlationData14MeV}, Length[#] > 0 &];

  (*Execute plot*)
  Print[ListPlot[plotData, Evaluate[plotOptions],
    PlotStyle → (Directive[PointSize[1 / 100], #] & /@ colors),
    PlotLegends → Placed[LineLegend[colors, labels, LabelStyle → 14], {0.82, 0.85}],
    Epilog → {Inset[Style[isotopeName, Bold, 15], Scaled@{0.15, 0.85}],
      Inset[Style[databaseName, Bold, 15], Scaled@{0.15, 0.90}]}]];




(*Define distance functions for each energy level*)
Module[{fitResult}, (*Define fitting and display functions*)
 fitAndPrint[correlationData_, energyLabel_] :=
  Module[{distanceFunction, coeffA, coeffB, coeffC, chargeNumber, normalizedFormula},
   Print[Style[energyLabel, FontSize → 16]];
   (*Fit with quadratic function*)
   distanceFunction = Fit[correlationData, {1, x, x^2}, x];
   (*Extract coefficients*){coeffC, coeffB, coeffA} =
```

```
    CoefficientList[distanceFunction, x];
   (*Calculate characteristic charge number*)kValue = -coeffA;
   chargeNumber = coeffB / kValue;
   mValue = coeffC;
   (*Display formula*)Print[Column[{Style[HoldForm[Reff] ==
        N[mValue, 6] + N[kValue, 6] * (N[chargeNumber, 6] - x) * x, FontSize → 16]}]];
   distanceFunction (*Return function*)];
 (*For 0.0253 eV case*)
 If[energyPattern == 1 || energyPattern == 2 || energyPattern == 3, distanceFunction0253eV =
   fitAndPrint[correlationData0253eV, "Incident Neutron Energy: 0.0253 eV case"]];
 (*For 500 keV case*)If[energyPattern == 2 || energyPattern == 3 ||
   energyPattern == 4 || energyPattern == 5, distanceFunction500keV =
   fitAndPrint[correlationData500keV, "Incident Neutron Energy: 500 keV case"]];
 (*For 14 MeV case*)If[energyPattern == 3 || energyPattern == 5, distanceFunction14MeV =
   fitAndPrint[correlationData14MeV, "Incident Neutron Energy: 14 MeV case"]];
 Print["\nAnalysis Results:"];
 Print["1. The effective fission distance
    Reff shows quadratic dependence on fragment charge number"];
 Print["2. This dependence reflects fundamental laws of charge
    distribution in the fission process"];
 Print["3. Similar dependence is maintained across different incident energies\n"];]



(*6*)(*Display and Analysis of Reff Values by Atomic Number*)
(*Output header for display*)Print[Style[
   "Table of Effective Fission Distance (Reff) Values by Atomic Number [Unit: fm]:",
   Bold, 16]];
Print[Style["Displaying calculated values (pre-fitting)
     and post-fitting values for each energy", 14]];
Print[Style["Fitting calculation used fragment values from atomic number " <>
    ToString[fitStartZ] <> " to " <> ToString[fitEndZ], 14]];



(*Create headers based on energy pattern*)
tableHeaders = Switch[energyPattern, 1,
   {"Z1", "Z2", "Reff [fm]\n(0.0253 eV)\nCalculated", "Fitted Value"},
   2, {"Z1", "Z2", "Reff [fm]\n(0.0253 eV)\nCalculated", "Fitted Value",
    "Reff [fm]\n(500 keV)\nCalculated", "Fitted Value"}, 3,
   {"Z1", "Z2", "Reff [fm]\n(0.0253 eV)\nCalculated", "Fitted Value",
    "Reff [fm]\n(500 keV)\nCalculated", "Fitted Value",
    "Reff [fm]\n(14 MeV)\nCalculated", "Fitted Value"}, 4,
   {"Z1", "Z2", "Reff [fm]\n(500 keV)\nCalculated", "Fitted Value"}, 5,
   {"Z1", "Z2", "Reff [fm]\n(500 keV)\nCalculated", "Fitted Value",
    "Reff [fm]\n(14 MeV)\nCalculated", "Fitted Value"}];

(*Function to get fitted value*)
getFittedDistanceValue[atomicNumber_, distanceFunction_] :=
  If[fitStartZ ≤ atomicNumber ≤ fitEndZ,
   NumberForm[N[distanceFunction /. x → atomicNumber], {6, 5}], "-"];

(*Create data table*)
tableData =
  Table[Module[{z1 = z, z2 = atomicNumber - z}, Flatten[{z1, z2, (*0.0253 eV data*)
```

```
      Which[energyPattern == 1 || energyPattern == 2 || energyPattern == 3,
       {NumberForm[N[effectiveDistance0253eV[z1, z2] /. fittedYield0253eV], {6, 6}],
        getFittedDistanceValue[z1, distanceFunction0253eV]}, True, {}],
      (*500 keV data*)Which[energyPattern == 2 || energyPattern == 3 ||
        energyPattern == 4 || energyPattern == 5,
       {NumberForm[N[effectiveDistance500keV[z1, z2] /. fittedYield500keV], {6, 6}],
        getFittedDistanceValue[z1, distanceFunction500keV]}, True, {}],
      (*14 MeV data*)Which[energyPattern == 3 || energyPattern == 5,
       {NumberForm[N[effectiveDistance14MeV[z1, z2] /. fittedYield14MeV], {6, 6}],
        getFittedDistanceValue[z1, distanceFunction14MeV]},
       True, {}]}]], {z, 23, 69}];



(*Output formatted table*)
Grid[Prepend[tableData, tableHeaders], Frame → All, Alignment → Center,
 Background → {None, {LightGray, None}}, ItemStyle → {Bold, "Text"},
 Dividers → {Join[{2}, Table[2 i + 2, {i, 1, Length[tableHeaders] / 2 - 1}]] → True,
   {2 → True}}, Spacings → {1.5, 1.2}]

(*7*)(*Generate Effective Fission Distance Functions*)
(*Generate functions for each energy region*)
If[energyPattern == 1 || energyPattern == 2 || energyPattern == 3,
  distanceData0253eV = Table[{x, atomicNumber - x} → distanceFunction0253eV,
    {x, fitStartZ - 6, fitEndZ + 6}]];

If[energyPattern == 2 || energyPattern == 3 || energyPattern == 4 || energyPattern == 5,
  distanceData500keV = Table[{x, atomicNumber - x} → distanceFunction500keV,
    {x, fitStartZ - 6, fitEndZ + 6}]];

If[energyPattern == 3 || energyPattern == 5, distanceData14MeV =
   Table[{x, atomicNumber - x} → distanceFunction14MeV, {x, fitStartZ - 6, fitEndZ + 6}]];

(*Define effective distance functions*)
If[energyPattern == 1 || energyPattern == 2 || energyPattern == 3, effDistanceFunc0253eV =
   Thread[Thread[effectiveDistance0253eV[distanceData0253eV[[All, 1, 1]],
       distanceData0253eV[[All, 1, 2]]]] → distanceData0253eV[[All, 2]]]];

If[energyPattern == 2 || energyPattern == 3 || energyPattern == 4 || energyPattern == 5,
  effDistanceFunc500keV =
   Thread[Thread[effectiveDistance500keV[distanceData500keV[[All, 1, 1]],
       distanceData500keV[[All, 1, 2]]]] → distanceData500keV[[All, 2]]]];

If[energyPattern == 3 || energyPattern == 5, effDistanceFunc14MeV =
   Thread[Thread[effectiveDistance14MeV[distanceData14MeV[[All, 1, 1]],
       distanceData14MeV[[All, 1, 2]]]] → distanceData14MeV[[All, 2]]]];

(*Calculate final yield data*)
If[energyPattern == 1 || energyPattern == 2 || energyPattern == 3,
  finalYield0253eV = fitYield0253eV /. effDistanceFunc0253eV];

If[energyPattern == 2 || energyPattern == 3 || energyPattern == 4 || energyPattern == 5,
  finalYield500keV = fitYield500keV /. effDistanceFunc500keV];

If[energyPattern == 3 || energyPattern == 5,
```

```
  finalYield14MeV = fitYield14MeV /. effDistanceFunc14MeV];

(*Display separator*)
Print[
  "--------------------------------------------------------------------------------
    -"];


(*8*)(*Comparison of Experimental and Theoretical Fission Yields*)
(*Display analysis summary*)
Print["Analysis of Fission Yields: Comparison
    between Experimental Data and Theoretical Calculations (Ex=0)"];
Print["\nTheoretical Analysis Results using Effective Fission
    Distance Reff proportional to fragment charge product,"];
Print["with zero Fermi Energy (Ex=0)"];
Print["- Quantitative reproduction of
    experimentally observed asymmetric fission yield distributions"];



(*Create common plot settings*)
commonPlotSettings = {Joined → {True, True},
   PlotRange → {{15, 80}, {10^(-12), 100}}, PlotMarkers → Automatic,
   Frame → True, FrameLabel → {"Atomic Number", "Fission Yield (Independent)
       "}, LabelStyle → Directive[Black, 19], FrameTicks → Automatic,
   FrameStyle → {Thick, Thick, Thick, Thick}, AspectRatio → 0.8, ImageSize → 450};

(*Legend settings*)
legendSettings = Placed[PointLegend[Automatic,
    {"JENDL-5 (Experimental)", "Theoretical curve (Fermi Energy=0)"},
    Joined → {True, True}, LabelStyle → Directive[Black, 15],
    LegendFunction → "Frame", LegendLayout → "Column",
    LegendMarkers → Array[{Graphics@Disk[], 10} &, 3]], {{0.26, 0.23}, {0.2, 0.9}}];


(*Plot style settings*)
plotStyles = {Directive[PointSize[1/100], Red],
   Directive[PointSize[1/100], Blue], Directive[PointSize[1/100], Green]};




(*Generate plots for each energy region*)
If[energyPattern == 1 || energyPattern == 2 || energyPattern == 3,
  Print["1. Analysis for Incident Neutron Energy: 0.0253eV"];
  Print[ListLogPlot[{yieldData0253eV[[fitStartIndex ;; fitEndIndex]],
     finalYield0253eV}, Evaluate[commonPlotSettings],
    PlotStyle → plotStyles, PlotLegends → legendSettings,
    Epilog → {Inset[Style[isotopeName, 20, Bold], Scaled@{0.16, 0.9}]}]];];

If[energyPattern == 2 || energyPattern == 3 || energyPattern == 4 || energyPattern == 5,
  Print["2. Analysis for Incident Neutron Energy: 500keV"];
  Print[ListLogPlot[{yieldData500keV[[fitStartIndex ;; fitEndIndex]],
     finalYield500keV}, Evaluate[commonPlotSettings],
```

```
    PlotStyle → plotStyles, PlotLegends → legendSettings,
    Epilog → {Inset[Style[isotopeName, 20, Bold], Scaled@{0.16, 0.9}]}]];];

If[energyPattern == 3 || energyPattern == 5,
  Print["3. Analysis for Incident Neutron Energy: 14MeV"];
  Print[ListLogPlot[{yieldData14MeV[[fitStartIndex ;; fitEndIndex]], finalYield14MeV},
    Evaluate[commonPlotSettings], PlotStyle → plotStyles, PlotLegends → legendSettings,
    Epilog → {Inset[Style[isotopeName, 20, Bold], Scaled@{0.16, 0.9}]}]];];
```

```
(*9*)(*Analysis of Effective Fission Distance and Fission Probability*)
(*Display section separator*)Print[
  "-------------------------------------------------------------------------------
    -"];
Print["Analysis of Effective Fission Distance and Fission Probability"];
Print[""];
Print["Analysis Contents:"];
Print["1. Calculation of Effective
    Fission Distance (Reff) for each incident neutron energy"];
Print["2. Evaluation of fission probability using η function"];
Print["3. Derivation of normalization factor κ (= Ex/Reff)"];
Print[""];
Print["The vertical axis κ represents the ratio of Fermi Energy(Ex)"];
Print["to effective fission distance (Reff)."];
Print[
  "κ ≈ 1 suggests the fission Fermi Energy is proportional to effective distance."];
Print[
  "-------------------------------------------------------------------------------
    -"];

(*Analysis and visualization module*)
Module[{}, (*Process for each incident neutron energy*)
  (*1. Calculate effective fission distance function values*)
  (*2. Calculate fission probability using η function*)
  (*3. Calculate normalization factor κ*)
  If[energyPattern == 1 || energyPattern == 2 || energyPattern == 3, distanceData0253eV =
    Table[{x, atomicNumber - x} → distanceFunction0253eV, {x, fitStartZ, fitEndZ}];
   distanceFunc0253eV = Thread[Thread[effectiveDistance0253eV[distanceData0253eV[[All,
         1, 1]], distanceData0253eV[[All, 1, 2]]]] → distanceData0253eV[[All, 2]]];
   normFactor0253eV = Thread[{distanceData0253eV[[All, 1, 1]],
      fittedYield0253eV[[All, 2]][[fitStartIndex ;; fitEndIndex]] /
       distanceFunc0253eV[[All, 2]]}];];
  If[energyPattern == 2 || energyPattern == 3 || energyPattern == 4 || energyPattern == 5,
   distanceData500keV =
    Table[{x, atomicNumber - x} → distanceFunction500keV, {x, fitStartZ, fitEndZ}];
   distanceFunc500keV = Thread[Thread[effectiveDistance500keV[distanceData500keV[[All,
         1, 1]], distanceData500keV[[All, 1, 2]]]] → distanceData500keV[[All, 2]]];
   normFactor500keV = Thread[{distanceData500keV[[All, 1, 1]],
      fittedYield500keV[[All, 2]][[fitStartIndex ;; fitEndIndex]] /
       distanceFunc500keV[[All, 2]]}];];
  If[energyPattern == 3 || energyPattern == 5, distanceData14MeV =
    Table[{x, atomicNumber - x} → distanceFunction14MeV, {x, fitStartZ, fitEndZ}];
```

```
 distanceFunc14MeV = Thread[Thread[effectiveDistance14MeV[distanceData14MeV[[All,
       1, 1]], distanceData14MeV[[All, 1, 2]]]] → distanceData14MeV[[All, 2]]];
 normFactor14MeV = Thread[{distanceData14MeV[[All, 1, 1]],
    fittedYield14MeV[[All, 2]][[fitStartIndex ;; fitEndIndex]] /
     distanceFunc14MeV[[All, 2]]}];];
(*Common plot options*)
plotOptions =
 {Joined → {True, True}, PlotRange → {{15, 80}, {0.9, 1.1}}, PlotMarkers → Automatic,
  Frame → True, FrameLabel → {"Atomic Number", "Normalization Factor κ = Ex/Reff"},
  LabelStyle → Directive[Black, 19], FrameTicks → Automatic,
  FrameStyle → {Thick, Thick, Thick, Thick}, AspectRatio → 1, ImageSize → 400};
(*Create visualization based on energy pattern*)
Switch[energyPattern, 1, Print[ListPlot[{normFactor0253eV},
   Evaluate[plotOptions], PlotStyle → {Directive[PointSize[1/100], Blue]},
   PlotLegends → Placed[PointLegend[{"0.0253eV"}, LabelStyle → 14,
      LegendFunction → "Frame"], {{0.975, 0.925}, {1, 0.9}}],
   Epilog → {Inset[Style[isotopeName, Bold, 14], Scaled@{0.15, 0.95}],
     Inset[Style[databaseName, Bold, 14], Scaled@{0.15, 0.90}]}]], 2, Print[
  ListPlot[{normFactor0253eV, normFactor500keV}, Evaluate[plotOptions], PlotStyle →
    {Directive[PointSize[1/100], Blue], Directive[PointSize[1/100], Green]},
   PlotLegends → Placed[PointLegend[{"0.0253eV", "500keV"}, LabelStyle → 14,
      LegendFunction → "Frame"], {{0.975, 0.925}, {1, 0.9}}],
   Epilog → {Inset[Style[isotopeName, Bold, 14], Scaled@{0.15, 0.95}],
     Inset[Style[databaseName, Bold, 14], Scaled@{0.15, 0.90}]}]], 3,
 Print[ListPlot[{normFactor0253eV, normFactor500keV, normFactor14MeV},
   Evaluate[plotOptions], PlotStyle → {Directive[PointSize[1/100], Blue],
     Directive[PointSize[1/100], Green], Directive[PointSize[1/100], Red]},
   PlotLegends → Placed[PointLegend[{"0.0253eV", "500keV", "14MeV"},
      LabelStyle → 14, LegendFunction → "Frame"], {{0.975, 0.925}, {1, 0.9}}],
   Epilog → {Inset[Style[isotopeName, Bold, 14], Scaled@{0.15, 0.95}],
     Inset[Style[databaseName, Bold, 14], Scaled@{0.15, 0.90}]}]],
 4, Print[ListPlot[{normFactor500keV}, Evaluate[plotOptions],
   PlotStyle → {Directive[PointSize[1/100], Green]},
   PlotLegends → Placed[PointLegend[{"500keV"}, LabelStyle → 14,
      LegendFunction → "Frame"], {{0.975, 0.925}, {1, 0.9}}],
   Epilog → {Inset[Style[isotopeName, Bold, 14], Scaled@{0.15, 0.95}],
     Inset[Style[databaseName, Bold, 14], Scaled@{0.15, 0.90}]}]], 5, Print[
  ListPlot[{normFactor500keV, normFactor14MeV}, Evaluate[plotOptions], PlotStyle →
    {Directive[PointSize[1/100], Green], Directive[PointSize[1/100], Red]},
   PlotLegends → Placed[PointLegend[{"500keV", "14MeV"}, LabelStyle → 14,
      LegendFunction → "Frame"], {{0.975, 0.925}, {1, 0.9}}],
   Epilog → {Inset[Style[isotopeName, Bold, 14], Scaled@{0.15, 0.95}],
     Inset[Style[databaseName, Bold, 14], Scaled@{0.15, 0.90}]}]]];];

Print[
  "-------------------------------------------------------------------------
    -"];



(*10*)(*Initialize Fermi Energy Variables and Main Calculation Function*)
(*Initialize Fermi Energy*)
```

```
fermiEnergy0253eV = .;
fermiEnergy500keV = .;
fermiEnergy14MeV = .;

(*Initialize parameter lists for each energy region*)
paramList0253eV = {};
paramList500keV = {};
paramList14MeV = {};

paramList5 = {};
paramList6 = {};
paramList7 = {};

(*Nuclear Fission Calculation Main Function*)
CalculateFissionYieldsWithfermi[energyPattern_] := Module[{tempData, results = {}},
  For[energyIndex = startEnergyIndex, energyIndex ≤ endEnergyIndex,
   energyIndex++, (*Set parameters for each energy region*)
   {promptNeutronCount, incidentEnergy, fermiEnergy, parameterList, yieldList,
     neutronVarList, effectiveDist, distanceFunc} = Switch[energyIndex, 1,
     {promptNeutrons1, neutronEnergy1, fermiEnergy0253eV, paramList0253eV,
      fissionYield0253eV, paramList5, effectiveDist0253eV, distanceFunction0253eV},
     (*Thermal*)2, {promptNeutrons2, neutronEnergy2, fermiEnergy500keV,
      paramList500keV, fissionYield500keV, paramList6, effectiveDist500keV,
      distanceFunction500keV}, (*Intermediate*)3, {promptNeutrons3,
      neutronEnergy3, fermiEnergy14MeV, paramList14MeV, fissionYield1400keV,
      paramList7, effectiveDist14MeV, distanceFunction14MeV}  (*Fast*)];
   tempData = Reap[For[protonNumber1 = 23, protonNumber1 ≤ atomicNumber - 23,
       protonNumber1++, For[neutronCount1 = 0, neutronCount1 ≤ neutronNumber,
        neutronCount1++, protonNumber2 = atomicNumber - protonNumber1;
        neutronCount2 = neutronNumber - neutronCount1 - Round[promptNeutronCount];
        massNumber1 = protonNumber1 + neutronCount1;
        massNumber2 = protonNumber2 + neutronCount2;


        (*Check fragment existence*)

        If[(protonNumber1 == 22 && 16 <= neutronCount1 <= 41) ||
          (protonNumber1 == 23 && 17 <= neutronCount1 <= 42) ||
          (protonNumber1 == 24 && 18 <= neutronCount1 <= 43) ||
          (protonNumber1 == 25 && 19 <= neutronCount1 <= 44) ||
          (protonNumber1 == 26 && 19 <= neutronCount1 <= 46) ||
          (protonNumber1 == 27 && 20 <= neutronCount1 <= 48) ||
          (protonNumber1 == 28 && 20 <= neutronCount1 <= 50) ||
          (protonNumber1 == 29 && 23 <= neutronCount1 <= 51) ||
          (protonNumber1 == 30 && 24 <= neutronCount1 <= 53) ||
          (protonNumber1 == 31 && 25 <= neutronCount1 <= 55) ||
          (protonNumber1 == 32 && 26 <= neutronCount1 <= 57) ||
          (protonNumber1 == 33 && 27 <= neutronCount1 <= 59) ||
          (protonNumber1 == 34 && 31 <= neutronCount1 <= 60) ||
          (protonNumber1 == 35 && 32 <= neutronCount1 <= 62) ||
          (protonNumber1 == 36 && 33 <= neutronCount1 <= 64) ||
          (protonNumber1 == 37 && 34 <= neutronCount1 <= 65) ||
          (protonNumber1 == 38 && 35 <= neutronCount1 <= 67) ||
```

```
  (protonNumber1 == 39 && 37 <= neutronCount1 <= 69) ||
  (protonNumber1 == 40 && 38 <= neutronCount1 <= 70) ||
  (protonNumber1 == 41 && 40 <= neutronCount1 <= 72) ||
  (protonNumber1 == 42 && 41 <= neutronCount1 <= 73) ||
  (protonNumber1 == 43 && 42 <= neutronCount1 <= 75) ||
  (protonNumber1 == 44 && 43 <= neutronCount1 <= 76) ||
  (protonNumber1 == 45 && 44 <= neutronCount1 <= 77) ||
  (protonNumber1 == 46 && 45 <= neutronCount1 <= 78) ||
  (protonNumber1 == 47 && 46 <= neutronCount1 <= 83) ||
  (protonNumber1 == 48 && 47 <= neutronCount1 <= 84) ||
  (protonNumber1 == 49 && 48 <= neutronCount1 <= 86) ||
  (protonNumber1 == 50 && 49 <= neutronCount1 <= 87) ||
  (protonNumber1 == 51 && 52 <= neutronCount1 <= 88) ||
  (protonNumber1 == 52 && 53 <= neutronCount1 <= 90) ||
  (protonNumber1 == 53 && 55 <= neutronCount1 <= 91) ||
  (protonNumber1 == 54 && 56 <= neutronCount1 <= 93) ||
  (protonNumber1 == 55 && 57 <= neutronCount1 <= 96) ||
  (protonNumber1 == 56 && 58 <= neutronCount1 <= 97) ||
  (protonNumber1 == 57 && 60 <= neutronCount1 <= 98) ||
  (protonNumber1 == 58 && 61 <= neutronCount1 <= 99) ||
  (protonNumber1 == 59 && 62 <= neutronCount1 <= 100) ||
  (protonNumber1 == 60 && 64 <= neutronCount1 <= 101) ||
  (protonNumber1 == 61 && 65 <= neutronCount1 <= 102) ||
  (protonNumber1 == 62 && 66 <= neutronCount1 <= 103) ||
  (protonNumber1 == 63 && 67 <= neutronCount1 <= 104) ||
  (protonNumber1 == 64 && 70 <= neutronCount1 <= 105) ||
  (protonNumber1 == 65 && 71 <= neutronCount1 <= 106) ||
  (protonNumber1 == 66 && 72 <= neutronCount1 <= 107) ||
  (protonNumber1 == 67 && 73 <= neutronCount1 <= 108) ||
  (protonNumber1 == 68 && 75 <= neutronCount1 <= 109) ||
  (protonNumber1 == 69 && 76 <= neutronCount1 <= 110) ||
  (protonNumber1 == 70 && 78 <= neutronCount1 <= 111) ||
  (protonNumber1 == 71 && 79 <= neutronCount1 <= 111) ||
  (protonNumber1 == 72 && 81 ≤ neutronCount1 ≤ 116) ||
  (protonNumber1 == 73 && 82 ≤ neutronCount1 ≤ 117) ||
  (protonNumber1 == 74 && 84 ≤ neutronCount1 ≤ 118) ||
  (protonNumber1 == 75 && 85 ≤ neutronCount1 ≤ 119),
 (*Check nuclear chart range for second fragment*)
 If[(protonNumber2 == 22 && 16 <= neutronCount2 <= 41) ||
   (protonNumber2 == 23 && 17 <= neutronCount2 <= 42) ||
   (protonNumber2 == 24 && 18 <= neutronCount2 <= 43) ||
   (protonNumber2 == 25 && 19 <= neutronCount2 <= 44) ||
   (protonNumber2 == 26 && 19 <= neutronCount2 <= 46) ||
   (protonNumber2 == 27 && 20 <= neutronCount2 <= 48) ||
   (protonNumber2 == 28 && 20 <= neutronCount2 <= 50) ||
   (protonNumber2 == 29 && 23 <= neutronCount2 <= 51) ||
   (protonNumber2 == 30 && 24 <= neutronCount2 <= 53) ||
   (protonNumber2 == 31 && 25 <= neutronCount2 <= 55) ||
   (protonNumber2 == 32 && 26 <= neutronCount2 <= 57) ||
   (protonNumber2 == 33 && 27 <= neutronCount2 <= 59) ||
```

```
 (protonNumber2 == 34 && 31 <= neutronCount2 <= 60) ||
 (protonNumber2 == 35 && 32 <= neutronCount2 <= 62) ||
 (protonNumber2 == 36 && 33 <= neutronCount2 <= 64) ||
 (protonNumber2 == 37 && 34 <= neutronCount2 <= 65) ||
 (protonNumber2 == 38 && 35 <= neutronCount2 <= 67) ||
 (protonNumber2 == 39 && 37 <= neutronCount2 <= 69) ||
 (protonNumber2 == 40 && 38 <= neutronCount2 <= 70) ||
 (protonNumber2 == 41 && 40 <= neutronCount2 <= 72) ||
 (protonNumber2 == 42 && 41 <= neutronCount2 <= 73) ||
 (protonNumber2 == 43 && 42 <= neutronCount2 <= 75) ||
 (protonNumber2 == 44 && 43 <= neutronCount2 <= 76) ||
 (protonNumber2 == 45 && 44 <= neutronCount2 <= 77) ||
 (protonNumber2 == 46 && 45 <= neutronCount2 <= 78) ||
 (protonNumber2 == 47 && 46 <= neutronCount2 <= 83) ||
 (protonNumber2 == 48 && 47 <= neutronCount2 <= 84) ||
 (protonNumber2 == 49 && 48 <= neutronCount2 <= 86) ||
 (protonNumber2 == 50 && 49 <= neutronCount2 <= 87) ||
 (protonNumber2 == 51 && 52 <= neutronCount2 <= 88) ||
 (protonNumber2 == 52 && 53 <= neutronCount2 <= 90) ||
 (protonNumber2 == 53 && 55 <= neutronCount2 <= 91) ||
 (protonNumber2 == 54 && 56 <= neutronCount2 <= 93) ||
 (protonNumber2 == 55 && 57 <= neutronCount2 <= 96) ||
 (protonNumber2 == 56 && 58 <= neutronCount2 <= 97) ||
 (protonNumber2 == 57 && 60 <= neutronCount2 <= 98) ||
 (protonNumber2 == 58 && 61 <= neutronCount2 <= 99) ||
 (protonNumber2 == 59 && 62 <= neutronCount2 <= 100) ||
 (protonNumber2 == 60 && 64 <= neutronCount2 <= 101) ||
 (protonNumber2 == 61 && 65 <= neutronCount2 <= 102) ||
 (protonNumber2 == 62 && 66 <= neutronCount2 <= 103) ||
 (protonNumber2 == 63 && 67 <= neutronCount2 <= 104) ||
 (protonNumber2 == 64 && 70 <= neutronCount2 <= 105) ||
 (protonNumber2 == 65 && 71 <= neutronCount2 <= 106) ||
 (protonNumber2 == 66 && 72 <= neutronCount2 <= 107) ||
 (protonNumber2 == 67 && 73 <= neutronCount2 <= 108) ||
 (protonNumber2 == 68 && 75 <= neutronCount2 <= 109) ||
 (protonNumber2 == 69 && 76 <= neutronCount2 <= 110) ||
 (protonNumber2 == 70 && 78 <= neutronCount2 <= 111) ||
 (protonNumber2 == 71 && 79 <= neutronCount2 <= 111) ||
 (protonNumber2 == 72 && 81 ≤ neutronCount2 ≤ 116) ||
 (protonNumber2 == 73 && 82 ≤ neutronCount2 ≤ 117) ||
 (protonNumber2 == 74 && 84 ≤ neutronCount2 ≤ 118) ||
 (protonNumber2 == 75 && 85 ≤ neutronCount2 ≤ 119),



Switch[energyIndex, 1, paramList5 = Union[AppendTo[paramList0253eV,
    fermiEnergy0253eV[protonNumber1, protonNumber2]]], 2,
 paramList6 = Union[AppendTo[paramList500keV, fermiEnergy500keV[
     protonNumber1, protonNumber2]]], 3, paramList7 = Union[AppendTo[
    paramList14MeV, fermiEnergy14MeV[protonNumber1, protonNumber2]]]];
(*Calculate fission parameters*)effectiveDistVal =
```

```
            distanceFunc /. x → protonNumber1;
           coulombEnergy = (1.44 * protonNumber1 * protonNumber2) / effectiveDistVal;
           qValue = (getNuclearMass[atomicNumber, atomicNumber + neutronNumber] -
               getNuclearMass[protonNumber1, massNumber1] -
               getNuclearMass[protonNumber2, massNumber2] -
               promptNeutronCount * 1.008665) * 931.4940954;
           effectiveEnergy = coulombEnergy - qValue;
           (*Calculate fission probability with fermi correction*)
           probability = 1 / (1 + Exp[2 * Pi / (neutronSeparationEnergy + incidentEnergy) *
                 Sqrt[protonNumber1 * protonNumber2 / (protonNumber1 + protonNumber2)] /
                  reducedMass * (effectiveEnergy -
                    fermiEnergy[protonNumber1, protonNumber2])]);
           Sow[{protonNumber1, probability}, yieldList];]]], {yieldList}, Rule][[
      2, All, 1]];
    (*Process results*)fragmentData = Part[yieldList /. tempData];
    processYields[data_] :=
     (Total@# / {Length@#, Total@data[[All, 2]] / 2} &) /@ GatherBy[data, First];
    AppendTo[results, {energyIndex, processYields[fragmentData]}];];
   results]

(*Execute calculation with Pattern selection*)
{startEnergyIndex, endEnergyIndex} =
  Switch[energyPattern, 1, {1, 1}, 2, {1, 2}, 3, {1, 3}, 4, {2, 2}, 5, {2, 3}];

(*Main calculation*)
results = CalculateFissionYieldsWithfermi[energyPattern];
{yieldData0253eVCalc, yieldData500keVCalc, yieldData14MeVCalc} =
  Switch[energyPattern, 1, {results[[1, 2]], Null, Null}, 2, {results[[1, 2]],
    results[[2, 2]], Null}, 3, {results[[1, 2]], results[[2, 2]], results[[3, 2]]}, 4,
   {Null, results[[1, 2]], Null}, 5, {Null, results[[1, 2]], results[[2, 2]]}];

(*11*)

(*Common Plot Generation Functions for Fission Yields*)
(*Define general yield plot function*)
CreateFissionYieldPlot[experimentalData_, calculatedData_, energyLabel_, plotColor_] :=
  ListLogPlot[{experimentalData, calculatedData}, (*Basic plot settings*)
   Joined → {True, True}, PlotRange → {{15, 80}, {10^(-12), 100}}, PlotMarkers →
    Automatic, (*Style settings*)PlotStyle → {Directive[PointSize[1 / 100], Red],
     Directive[PointSize[1 / 100], plotColor]}, (*Frame settings*)
   Frame → True, FrameLabel → {"Atomic Number", "Fission Yield (Independent)"},
   LabelStyle → Directive[Black, 19], FrameTicks → Automatic,
   FrameStyle → {Thick, Thick, Thick, Thick}, (*Legend settings*)
   PlotLegends → Placed[PointLegend[Automatic, {"JENDL-5", "Theoretical Curve"},
      Joined → {True, True, True}, Joined → {True, True},
      LabelStyle → 16, LegendFunction → "Frame", LegendLayout → "Column",
      LegendMarkers → Array[{Graphics@Disk[], 10} &, 3]], {{0.72, 0.25}, {1, 0.9}}],
   (*Layout settings*)AspectRatio → 0.8, ImageSize → 400, (*Title and isotope
    label*)Epilog → Inset[Style[isotopeName, Bold, 18], Scaled@{0.14, 0.94}]];

(*Define data analysis function*)
AnalyzeIsotopeYield[yieldData_, label_] :=
  Module[{maxZ1, maxZ2, maxYield1, maxYield2, peakAvgYield},
   (*Display analysis header*)Print[label];
```

```
    (*Find primary peak*)
    maxZ1 = Position[yieldData, Max[yieldData[[All, 2]]]][[1, 1]] + fitStartZ - 1;
    maxZ2 = atomicNumber - maxZ1;
    maxYield1 = Max[yieldData[[All, 2]]];
    maxYield2 = yieldData[[maxZ2 - fitStartZ + 1]][[2]];
    (*Display isotope information*)
    For[atomicNum = 23, atomicNum ≤ 71, atomicNum++,
     If[maxZ1 == atomicNum, Print["Primary Fragment: ", ElementData[atomicNum, "Name"],
       "(Z=", atomicNum, ")", "; Yield: ", maxYield1, " MeV"]];
     If[maxZ2 == atomicNum, Print["Secondary Fragment: ", ElementData[atomicNum, "Name"],
       "(Z=", atomicNum, ")", "; Yield: ", maxYield2, " MeV"]];];
    (*Calculate and display average peak yield*)
    peakAvgYield = (maxYield1 + maxYield2) / 2;
    Print["Average Peak Yield: ", peakAvgYield, " MeV"];
    Print[
     "-------------------------------------------------------------------------
       ---"];];


(*12*)(*Optimization Program for Fermi Energy*)(*Display program description*)
Print[
  "Theoretical Analysis and Experimental Comparison of Fission Yield Distributions"];
Print[""];
Print["Calculation Process:"];
Print["1. Evaluate logarithmic differences between
    JENDL-5 experimental data and theoretical calculations"];
Print["2. Optimize Fermi Energy (Ex) using least squares method"];
Print["3. Generate theoretical curves using optimized Ex"];
Print["4. Compare and verify experimental vs theoretical values"];
Print[""];
Print["Optimization Goals:"];
Print[
  " · Theoretical reproduction of experimentally observed asymmetric fission yields"];
Print[" · Understanding fission mechanisms at each incident neutron energy"];
Print[" · Systematic determination of Fermi Energy (Ex)"];
Print[""];
Print["Evaluation Methods:"];
Print[
  " · Minimize sum of squared logarithmic differences between theory and experiment"];
Print[" · Parameter optimization using FindMinimum function"];
Print[" · Validation of theoretical curves with optimized parameters"];
Print[
  "-------------------------------------------------------------------------
    -"];




(*Main optimization loop for each energy region*)
For[energyRegion = startEnergyIndex, energyRegion ≤ endEnergyIndex, energyRegion++,



  (*Initialize variables for each energy region*)
  If[energyRegion == 1,
```

```
 fermiParams0253eV = Union[paramList5] (*Thermal neutron parameters*)];
If[energyRegion == 2, fermiParams500keV = Union[paramList6]
 (*Intermediate energy parameters*)];
If[energyRegion == 3, fermiParams14MeV = Union[paramList7]
 (*Fast neutron parameters*)];


(*Prepare experimental data*)
If[energyRegion == 1,
 expData = yieldData0253eV[[fitStartIndex ;; fitEndIndex]] (*JENDL-5 0.0253eV*)];
If[energyRegion == 2, expData = yieldData500keV[[fitStartIndex ;; fitEndIndex]]
 (*JENDL-5 500keV*)];
If[energyRegion == 3, expData = yieldData14MeV[[fitStartIndex ;; fitEndIndex]]
 (*JENDL-5 14MeV*)];



(*Prepare theoretical calculation data*)
If[energyRegion == 1,
 yieldData0253eVTheory = yieldData0253eVCalc[[fitStartIndex ;; fitEndIndex]]];
If[energyRegion == 2, yieldData500keVTheory =
  yieldData500keVCalc[[fitStartIndex ;; fitEndIndex]]];
If[energyRegion == 3, yieldData14MeVTheory =
  yieldData14MeVCalc[[fitStartIndex ;; fitEndIndex]]];




(*Store optimized parameters*)
If[energyRegion == 1, fittedParams0253eV = optResult0253eVRe2[[2, All, All]]];
If[energyRegion == 2, fittedParams500keV = optResult500keVRe2[[2, All, All]]];
If[energyRegion == 3, fittedParams14MeV = optResult14MeVRe2[[2, All, All]]];
(*Display optimized parameters*)
If[energyRegion == 1, Print[fittedParams0253eV[[fitStartIndex ;; fitEndIndex]]]];
If[energyRegion == 2, Print[fittedParams500keV[[fitStartIndex ;; fitEndIndex]]]];
If[energyRegion == 3, Print[fittedParams14MeV[[fitStartIndex ;; fitEndIndex]]]];
(*Calculate theoretical yields with optimized parameters*)
If[energyRegion == 1, theoreticalYield0253eV =
  yieldData0253eVTheory /. optResult0253eVRe2[[2, All, All]]];
If[energyRegion == 2, theoreticalYield500keV =
  yieldData500keVTheory /. optResult500keVRe2[[2, All, All]]];
If[energyRegion == 3, theoreticalYield14MeV =
  yieldData14MeVTheory /. optResult14MeVRe2[[2, All, All]]];
(*Generate and display plots for each energy region*)
If[energyRegion == 1, Print["1. Analysis for Incident Neutron Energy: 0.0253eV"];
 Print[CreateFissionYieldPlot[expData,
   theoreticalYield0253eV, "0.0253eV", Blue]];];
If[energyRegion == 2, Print["2. Analysis for Incident Neutron Energy: 500keV"];
 Print[CreateFissionYieldPlot[expData, theoreticalYield500keV, "500keV", Blue]];];
If[energyRegion == 3, Print["3. Analysis for Incident Neutron Energy: 14MeV"];
 Print[CreateFissionYieldPlot[expData, theoreticalYield14MeV, "14MeV", Blue]];];
(*Display separator*)Print[
 "-------------------------------------------------------------------------
   ---"];];
```

```
(*13*)(*Analysis and Visualization of Optimization Results*)
(*Display analysis title*)
Print["Quantitative Analysis Results of Fermi Energy Ex"];
Print["Based on Optimization Calculations using Experimental Fission Yield Data"];

(*Process and display results for each energy region*)
If[energyPattern == 1 || energyPattern == 2 || energyPattern == 3,
  Print["1. Analysis for Incident Neutron Energy: 0.0253eV"];
  Print[fermiData0253eV = Thread[{fittedParams0253eV[[All, 1, 1]],
       fittedParams0253eV[[All, 2]]}][[fitStartIndex ;; fitEndIndex]]];];

If[energyPattern == 2 || energyPattern == 3 || energyPattern == 4 || energyPattern == 5,
  Print["2. Analysis for Incident Neutron Energy: 500keV"];
  Print[fermiData500keV = Thread[{fittedParams500keV[[All, 1, 1]],
       fittedParams500keV[[All, 2]]}][[fitStartIndex ;; fitEndIndex]]];];

If[energyPattern == 3 || energyPattern == 5,
  Print["3. Analysis for Incident Neutron Energy: 14MeV"];
  Print[fermiData14MeV = Thread[{fittedParams14MeV[[All, 1, 1]],
       fittedParams14MeV[[All, 2]]}][[fitStartIndex ;; fitEndIndex]]];];

Print[
  "-------------------------------------------------------------------------
    -"];


Print[" Systematic Analysis Results of Fermi Energy Ex"];
Print[" Using Optimized Effective Fission Distance Reff"];


(*Define common plot settings*)
plotBaseSettings = {PlotRange → {{15, 75}, {-8, 12}}, PlotMarkers → Automatic,
   Frame → True, FrameLabel → {"Atomic Number", "Energy (MeV)"},
   LabelStyle → Directive[Black, 19], FrameTicks → Automatic,
   FrameStyle → {Thick, Thick, Thick, Thick}, AspectRatio → 1.1, ImageSize → 400};


(*Create visualization based on energy pattern*)
Switch[energyPattern, 1, (*0.0253eV only*)plotData = {fermiData0253eV, {}, {}};
  Print[ListPlot[Select[plotData, Length[#] > 0 &],
    Joined → {True, True}, Evaluate@plotBaseSettings, PlotStyle →
     {Directive[PointSize[1/100], Blue], Directive[PointSize[1/100], Green],
      Directive[PointSize[1/100], Red]}, PlotLegends →
     Placed[PointLegend[Automatic, {"0.0253eV"}, Joined → {True, True, True},
       LabelStyle → 14, LegendFunction → "Frame", LegendLayout → "Column",
       LegendMarkers → Array[{Graphics@Disk[], 10} &, 3]], {{0.955, 0.95}, {1, 0.9}}],
    Epilog → {Inset[Style[isotopeName, 20, Bold], Scaled@{0.15, 0.95}],
      Inset[Style[databaseName, 20, Bold], Scaled@{0.15, 0.90}]}]], 2,
  (*0.0253eV and 500keV*)plotData = {fermiData0253eV, fermiData500keV, {}};
  Print[ListPlot[Select[plotData, Length[#] > 0 &],
    Joined → {True, True}, Evaluate@plotBaseSettings, PlotStyle →
     {Directive[PointSize[1/100], Blue], Directive[PointSize[1/100], Green],
      Directive[PointSize[1/100], Red]}, PlotLegends →
     Placed[PointLegend[Automatic, {"0.0253eV", "500keV"}, Joined → {True, True, True},
```

```
        LabelStyle → 14, LegendFunction → "Frame", LegendLayout → "Column",
        LegendMarkers → Array[{Graphics@Disk[], 10} &, 3]], {{0.955, 0.95}, {1, 0.9}}],
      Epilog → {Inset[Style[isotopeName, 20, Bold], Scaled@{0.15, 0.95}],
        Inset[Style[databaseName, 20, Bold], Scaled@{0.15, 0.90}]}]],
    3, (*All three energies*)plotData = {fermiData0253eV,
      fermiData500keV, fermiData14MeV};
    Print[ListPlot[Select[plotData, Length[#] > 0 &], Joined → {True, True},
      Evaluate@plotBaseSettings, PlotStyle → {Directive[PointSize[1/100], Blue],
        Directive[PointSize[1/100], Green], Directive[PointSize[1/100], Red]},
      PlotLegends → Placed[PointLegend[Automatic, {"0.0253eV", "500keV", "14MeV"},
        Joined → {True, True, True}, LabelStyle → 14,
        LegendFunction → "Frame", LegendLayout → "Column",
        LegendMarkers → Array[{Graphics@Disk[], 10} &, 3]], {{0.955, 0.95}, {1, 0.9}}],
      Epilog → {Inset[Style[isotopeName, 20, Bold], Scaled@{0.15, 0.95}],
        Inset[Style[databaseName, 20, Bold], Scaled@{0.15, 0.90}]}]],
    4, (*500keV only*)plotData = {fermiData500keV};
    Print[ListPlot[plotData, Joined → True, PlotRange → {{20, 75}, {-8, 12}},
      PlotMarkers → Automatic, PlotStyle → Directive[PointSize[1/100], Blue],
      Frame → True, LabelStyle → Directive[Black, 19],
      FrameTicks → Automatic, FrameStyle → {Thick, Thick, Thick, Thick},
      PlotLegends → Placed[PointLegend[{Blue}, {"500keV"}, Joined → True,
        LabelStyle → 14, LegendFunction → "Frame", LegendLayout → "Column",
        LegendMarkers → {Graphics@Disk[]}], {{0.955, 0.95}, {1, 0.9}}], AspectRatio →
       1.1, Epilog → {Inset[Style[isotopeName, 20, Bold], Scaled@{0.15, 0.95}],
        Inset[Style[databaseName, 20, Bold], Scaled@{0.15, 0.90}]}]], 5,
    (*500keV and 14MeV*)plotData = {{}, fermiData500keV, fermiData14MeV};
    Print[ListPlot[Select[plotData, Length[#] > 0 &], Joined → {True, True},
      PlotRange → {{20, 75}, {-8, 12}}, PlotMarkers → Automatic, PlotStyle →
       {Directive[PointSize[1/100], Green], Directive[PointSize[1/100], Red]},
      Frame → True, LabelStyle → Directive[Black, 19], FrameTicks → Automatic,
      FrameStyle → {Thick, Thick, Thick, Thick},
      PlotLegends → Placed[PointLegend[Automatic, {"500keV", "14MeV"},
        Joined → {True, True}, LabelStyle → 14, LegendFunction → "Frame",
        LegendLayout → "Column", LegendMarkers → Array[{Graphics@Disk[], 10} &, 3]],
       {{0.955, 0.95}, {1, 0.9}}], AspectRatio → 1.1,
      Epilog → {Inset[Style[isotopeName, 20, Bold], Scaled@{0.15, 0.95}],
        Inset[Style[databaseName, 20, Bold], Scaled@{0.15, 0.90}]}]]];
  (*14*)(*Final Analysis and Visualization of Nuclear Species*)
  (*Fragment Analysis Function*)
  AnalyzeIsotopeYield[yieldData_, label_] := Module[{centerZ = Floor[atomicNumber/2],
      (*Calculate center atomic number*)rangeStart = Floor[atomicNumber/2] - 10,
      (*Lower bound*)rangeEnd = Floor[atomicNumber/2] + 10, (*Upper bound*)
      maxZ1, maxZ2, maxYield1, maxYield2, peakAvgYield, filteredData},
     (*Display analysis header*)Print[label];
     (*Filter data to only include atomic numbers within our range*)
     filteredData = Select[yieldData, rangeStart ≤ First[#] ≤ rangeEnd &];
     (*Find maximum yield within our range*)maxYield1 = Max[filteredData[[All, 2]]];
     maxZ1 = First[First[Select[filteredData, #[[2]] == maxYield1 &]]];
     maxZ2 = atomicNumber - maxZ1;
     maxYield2 = yieldData[[maxZ2 - First[yieldData][[1]] + 1, 2]];
     (*Display fragment information*)Print["Secondary Fragment: ",
      ElementData[maxZ1, "Name"], "(Z=", maxZ1, "); Yield: ", maxYield1, " MeV"];
     Print["Primary Fragment: ", ElementData[maxZ2, "Name"],
      "(Z=", maxZ2, "); Yield: ", maxYield2, " MeV"];
```

```
    (*Calculate and display average peak yield*)
    peakAvgYield = (maxYield1 + maxYield2) / 2;
    Print["Average Peak Yield: ", peakAvgYield, " MeV"];
    Print[
     "------------------------------------------------------------------------
       ---"];];

(*Perform isotope analysis for each energy region*)
If[energyPattern == 1 || energyPattern == 2 || energyPattern == 3, AnalyzeIsotopeYield[
   fermiData0253eV, "1. Analysis for Incident Neutron Energy: 0.0253eV"]];

If[energyPattern == 2 || energyPattern == 3 || energyPattern == 4 || energyPattern == 5,
  AnalyzeIsotopeYield[fermiData500keV,
   "2. Analysis for Incident Neutron Energy: 500keV"]];

If[energyPattern == 3 || energyPattern == 5, AnalyzeIsotopeYield[
   fermiData14MeV, "3. Analysis for Incident Neutron Energy: 14MeV"];
  Print[
   "------------------------------------------------------------------------
     ---"]];

(*Define logarithmic scale transformation function*)
LogScaleTransform[value_] := Rescale[Log10[value], {-8, 1}, {-8, 12}];

(*Common plot settings for combined visualization*)
combinedPlotSettings =
  {Joined → {True}, PlotRange → {{15, 75}, {-10, 12}}, PlotMarkers → Automatic,
   Frame → True, FrameLabel → {{Style["Energy (MeV)", 19, Black],
      Style["Fission Yield (Independent)", 19, Black]},
     {Style["Atomic Number", 19, Black], None}},
   LabelStyle → Directive[Black, 19], FrameTicks →
    {{Automatic, ({LogScaleTransform[10^#], If[# == 0, 1, Superscript[10, #]]} &) /@
       {-8, -6, -4, -2, 0, 2}}, {Automatic, None}},
   FrameStyle → {Thick, Thick, Thick, Thick}, AspectRatio → 1.1,
   ImageSize → 400, Axes → {True, False}};

(*Plot generation function*)
CreateEnergyPlot[fermiData_, yieldData_, energy_, color_] := Module[{scaledYieldData},
   scaledYieldData = ({#[[1]], LogScaleTransform@#[[2]]} &) /@ yieldData;
   ListPlot[{fermiData, scaledYieldData[[fitStartIndex ;; fitEndIndex]]},
    Evaluate[combinedPlotSettings],
    PlotStyle → {Directive[PointSize[1 / 100], color], Directive[PointSize[1 / 100],
       GrayLevel[0.6 - 0.2 * Position[{Blue, Green, Red}, color][[1, 1]]]]},
    PlotLegends → Placed[PointLegend[Automatic, {"Fermi Energy",
        "Charge Distribution (" <> databaseName <> ")"}, Joined → True,
       LabelStyle → 14, LegendFunction → "Frame", LegendLayout → "Column",
       LegendMarkers → Array[{Graphics@Disk[], 10} &, 2]], {{0.94, 0.19}, {1, 0.9}}],
    Epilog → {Inset[Style[ToString[isotopeName] <> " " <> energy, 18, Bold],
       Scaled@{0.4, 0.94}]}]];

(*Generate plots for each energy region*)
Module[{}, If[energyPattern == 1 || energyPattern == 2 || energyPattern == 3,
   Print[CreateEnergyPlot[fermiData0253eV, yieldData0253eV, "0.0253eV", Blue]]];
  If[energyPattern == 2 || energyPattern == 3 || energyPattern == 4 || energyPattern == 5,
```

```
    Print[CreateEnergyPlot[fermiData500keV, yieldData500keV, "500keV", Green]]];
   If[energyPattern == 3 || energyPattern == 5,
    Print[CreateEnergyPlot[fermiData14MeV, yieldData14MeV, "14MeV", Red]]];];

(*Final separator*)
Print[
  "-------------------------------------------------------------------------------
    -"];
```