StefinRacho-Homework-2

September 18, 2024

0.1 Homework 2

0.1.1 Due Friday, 9/20/2024

0.1.2 For each homework set:

0.1.3 Create a New iPython (Jupyter) notebook. Name the notebook FirstAndLastName_Homework2 and save it before you start working

0.1.4 To submit, export or print your notebook as a pdf, with all outputs visible. Upload both the pdf and a copy of your notebook (.ipynb) in Canvas.

1. Exercise 2.8: Create two arrays a and b a = [1,2,3,4] b = [2,4,6,8] Compute the following:

   b/a + 1 b/(a+1) 1/a

   Verify these values by computing it by hand. (Either paper or using Markdown)

[3]:
```python
import numpy as np
import timeit
```

[30]:
```python
a = np.array([1, 2, 3, 4])
b = np.array([2, 4, 6, 8])
print(f"a = {a}")
print(f"b = {b}")
print(f"b/a + 1 = {b/a + 1}")
print(f"b/(a+1) = {b/(a+1)}")
print(f"1/a = {1/a}")
```

```
a = [1 2 3 4]
b = [2 4 6 8]
b/a + 1 = [3. 3. 3. 3.]
b/(a+1) = [1.         1.33333333 1.5        1.6       ]
1/a = [1.         0.5        0.33333333 0.25      ]
```

Solving for b/a + 1: b/a + 1 = [2/1 + 1, 4/2 + 1, 6/3 + 1, 8/4 + 1] b/a + 1 = [2 + 1, 2 + 1, 2 + 1, 2 + 1] b/a + 1 = [3, 3, 3, 3]

Solving for b/(a+1): b/(a+1) = [2/(1+1), 4/(2+1), 6/(3+1), 8/(4+1)] b/(a+1) = [2/2, 4/3, 6/4, 8/5] b/(a+1) = [1, 1.3333, 1.5, 1.6]

Solving for 1/a: 1/a = [1/1, 1/2, 1/3, 1/4] 1/a = [1, 0.5, 0.3333, 0.25]

2. Using the same arrays above. Write a program to compute the dot product between them by accessing the elements of the array individually. Verify the result using the built-in function. (np.dot) You may need to look up what a dot product is.

[90]:
```
def dotProduct(arr1, arr2):
    product = 0
    for i in np.arange(arr1.size):
        print(f"arr1[{i}] = {arr1[i]}\narr2[{i}] = {arr2[i]}")
        print(f"{arr1[i]} * {arr2[i]} = {arr1[i] * arr2[i]}")
        print(f"{product} + {arr1[i] * arr2[i]} = {product + arr1[i] * arr2[i]}")
        product += arr1[i] * arr2[i]
        print(f"dot product so far is {product}\n")
    return product

dotProduct(a, b)
print(f"np.dot = {np.dot(a, b)}")
```

```
arr1[0] = 1
arr2[0] = 2
1 * 2 = 2
0 + 2 = 2
dot product so far is 2

arr1[1] = 2
arr2[1] = 4
2 * 4 = 8
2 + 8 = 10
dot product so far is 10

arr1[2] = 3
arr2[2] = 6
3 * 6 = 18
10 + 18 = 28
dot product so far is 28

arr1[3] = 4
arr2[3] = 8
4 * 8 = 32
28 + 32 = 60
dot product so far is 60

np.dot = 60
```

3. Download the Gaussian.txt found on Canvas and put it in the same directory as your code. Load the values from the file using numpy.loadtxt() Compute the following: the sum, length, mean of all the values in the file. Then compute root mean square deviation (RMSD)
$RMSD = \sqrt{\frac{\sum_{i=0}^{i=N}(mean-values[i])^2}{N}}$
Since the values in the Gaussian.txt is drawn from a Gaussian distribution. The mean and

RMSD should be almost the same as a Gaussian centered at 10 and with a sigma of 2.

[10]:
```python
x = np.loadtxt("Gaussian.txt")
sum = np.sum(x)
print(f"sum = {sum}")

length = x.size
print(f"length = {length}")

mean = np.mean(x)
print(f"mean = {mean}")

RMSD = np.sqrt(np.sum((x - mean)**2) / length)
print(f"RMSD = {RMSD}")
```

```
sum = 10037.52371
length = 1000
mean = 10.03752371
RMSD = 1.922871400919036
```

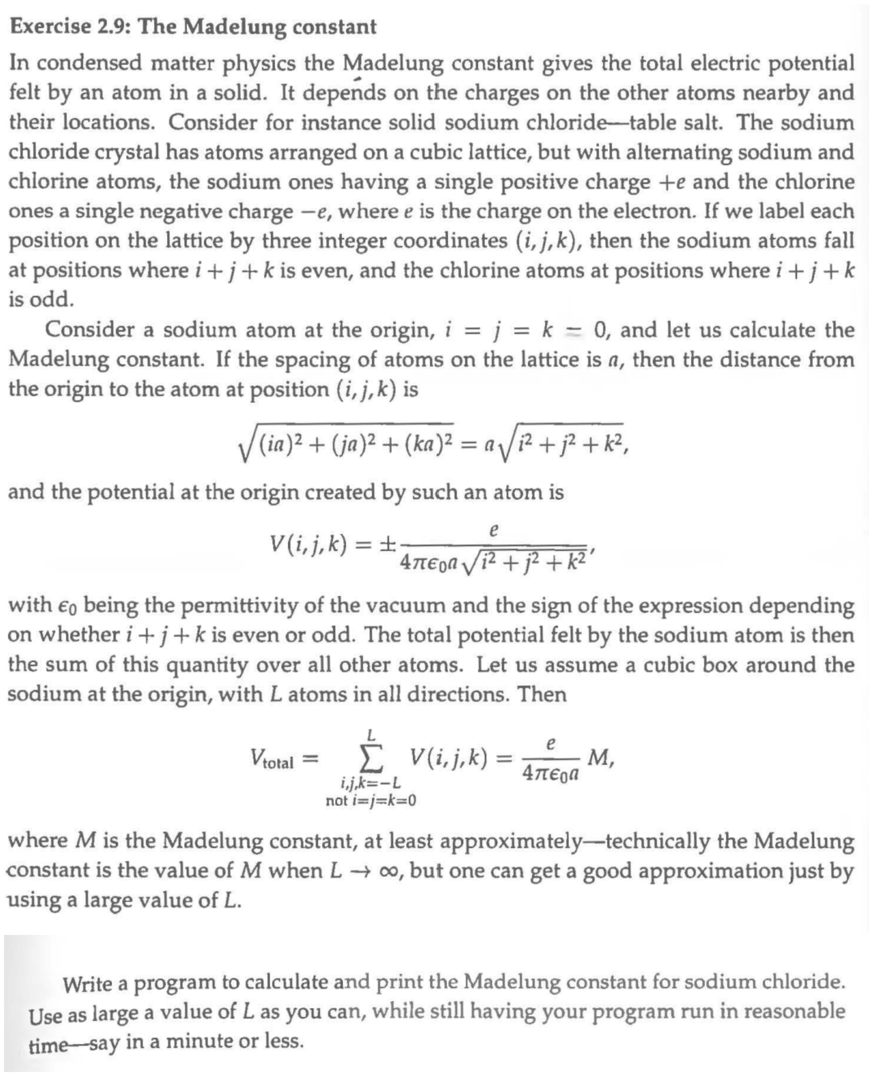
4. Download the matrix.txt found on Canvas and put in the same directory as your code
   - Load the values from the file using numpy.loadtxt()
   - print out the following properties of the matrix and explain what each tell you about the array:
     - len, shape, sum, min, max (note that some of these function will require you loop through each row of the matrix)
   - Slice the matrix to print out only the 1st Row
   - Slice the matrix to print out only the 1st Column
   - Slice the matrix to print out only the 1st 3 Row
   - Slice the matrix to print out only the 1st 3 Column

   - Slice the matrix to print out only the last 3 Row
   - Slice the matrix to print out only the last 3 Column

[27]:
```python
matrix = np.loadtxt("matrix.txt")
print(f"matrix = \n{matrix}\n")

print(f"matrix length = {matrix.size}\nLength describes how many elements are in␣
 ↪the matrix.\n")
print(f"matrix shape = {matrix.shape}\nShape describes the dimensions of the␣
 ↪array.\n")
print(f"matrix sum = {matrix.sum()}\nSum describes the value when all the␣
 ↪elements in the matrix are added together.\n")
print(f"matrix min = {matrix.min()}\nMin describes the minimum value in the␣
 ↪matrix.\n")
print(f"matrix max = {matrix.max()}\nMax describes the maximum value in the␣
 ↪matrix.\n")
print(f"1st Row = {matrix[0, :]}\n")
print(f"1st Column = {matrix[:, 0]}\n")
print(f"First 3 Rows =\n{matrix[:3, :]}\n")
print(f"First 3 Columns =\n{matrix[:, :3]}\n")
print(f"Last 3 Rows =\n{matrix[-3:, :]}\n")
print(f"Last 3 Columns=\n{matrix[:, -3:]}\n")
```

```
matrix =
[[ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9.]
 [10. 11. 12. 13. 14. 15. 16. 17. 18. 19.]
 [20. 21. 22. 23. 24. 25. 26. 27. 28. 29.]
 [30. 31. 32. 33. 34. 35. 36. 37. 38. 39.]
 [40. 41. 42. 43. 44. 45. 46. 47. 48. 49.]
 [50. 51. 52. 53. 54. 55. 56. 57. 58. 59.]
 [60. 61. 62. 63. 64. 65. 66. 67. 68. 69.]
 [70. 71. 72. 73. 74. 75. 76. 77. 78. 79.]
 [80. 81. 82. 83. 84. 85. 86. 87. 88. 89.]
 [90. 91. 92. 93. 94. 95. 96. 97. 98. 99.]]

matrix length = 100
Length describes how many elements are in the matrix.

matrix shape = (10, 10)
Shape describes the dimensions of the array.

matrix sum = 4950.0
Sum describes the value when all the elements in the matrix are added together.

matrix min = 0.0
Min describes the minimum value in the matrix.

matrix max = 99.0
Max describes the maximum value in the matrix.

1st Row = [0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]

1st Column = [ 0. 10. 20. 30. 40. 50. 60. 70. 80. 90.]

First 3 Rows =
[[ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9.]
 [10. 11. 12. 13. 14. 15. 16. 17. 18. 19.]
 [20. 21. 22. 23. 24. 25. 26. 27. 28. 29.]]

First 3 Columns =
[[ 0.  1.  2.]
 [10. 11. 12.]
 [20. 21. 22.]
 [30. 31. 32.]
 [40. 41. 42.]
 [50. 51. 52.]
 [60. 61. 62.]
 [70. 71. 72.]
 [80. 81. 82.]
 [90. 91. 92.]]

Last 3 Rows =
[[70. 71. 72. 73. 74. 75. 76. 77. 78. 79.]
 [80. 81. 82. 83. 84. 85. 86. 87. 88. 89.]
 [90. 91. 92. 93. 94. 95. 96. 97. 98. 99.]]

Last 3 Columns=
[[ 7.  8.  9.]
 [17. 18. 19.]
 [27. 28. 29.]
 [37. 38. 39.]
 [47. 48. 49.]
 [57. 58. 59.]
 [67. 68. 69.]
 [77. 78. 79.]
 [87. 88. 89.]
 [97. 98. 99.]]
```

5. Exercise 2.9



Exercise 2.9: The Madelung constant

In condensed matter physics the Madelung constant gives the total electric potential felt by an atom in a solid. It depends on the charges on the other atoms nearby and their locations. Consider for instance solid sodium chloride—table salt. The sodium chloride crystal has atoms arranged on a cubic lattice, but with alternating sodium and chlorine atoms, the sodium ones having a single positive charge $+e$ and the chlorine ones a single negative charge $-e$, where e is the charge on the electron. If we label each position on the lattice by three integer coordinates (i, j, k), then the sodium atoms fall at positions where $i + j + k$ is even, and the chlorine atoms at positions where $i + j + k$ is odd.

Consider a sodium atom at the origin, $i = j = k = 0$, and let us calculate the Madelung constant. If the spacing of atoms on the lattice is a, then the distance from the origin to the atom at position (i, j, k) is

$$\sqrt{(ia)^2 + (ja)^2 + (ka)^2} = a\sqrt{i^2 + j^2 + k^2},$$

and the potential at the origin created by such an atom is

$$V(i, j, k) = \pm \frac{e}{4\pi\epsilon_0 a\sqrt{i^2 + j^2 + k^2}},$$

with ϵ_0 being the permittivity of the vacuum and the sign of the expression depending on whether $i + j + k$ is even or odd. The total potential felt by the sodium atom is then the sum of this quantity over all other atoms. Let us assume a cubic box around the sodium at the origin, with L atoms in all directions. Then

$$V_{\text{total}} = \sum_{\substack{i,j,k=-L \\ \text{not } i=j=k=0}}^{L} V(i, j, k) = \frac{e}{4\pi\epsilon_0 a} M,$$

where M is the Madelung constant, at least approximately—technically the Madelung constant is the value of M when $L \to \infty$, but one can get a good approximation just by using a large value of L.

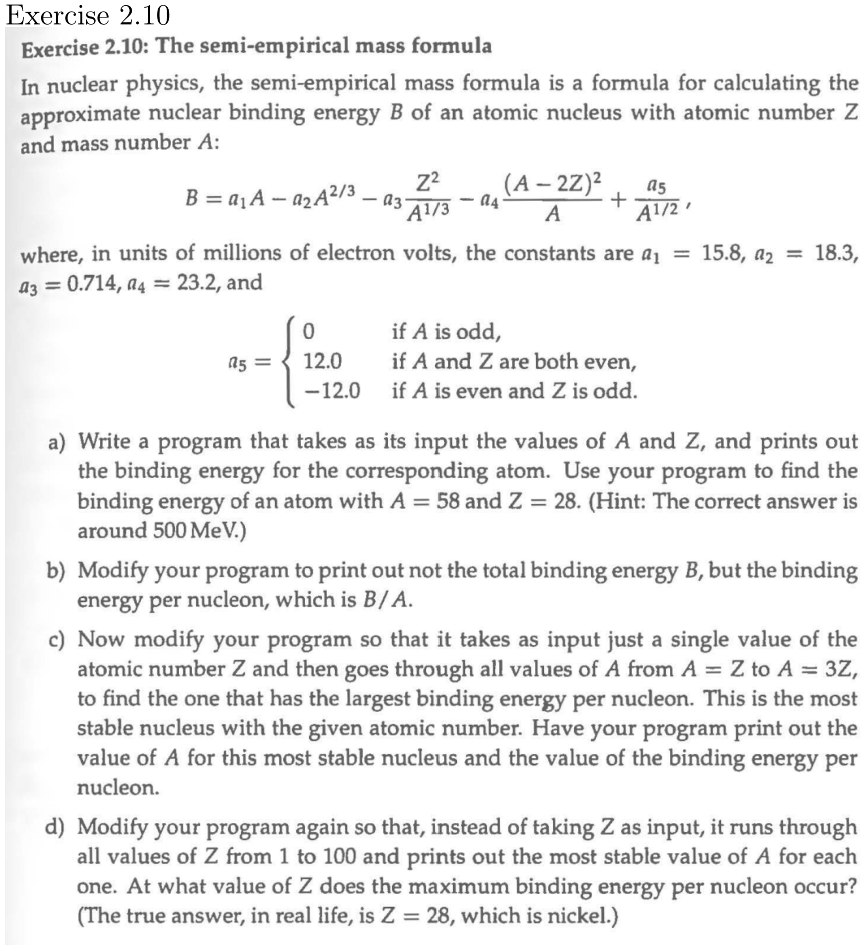
Write a program to calculate and print the Madelung constant for sodium chloride. Use as large a value of L as you can, while still having your program run in reasonable time—say in a minute or less.

[11]:
```python
L = 325
start_time = timeit.default_timer()

i, j, k = np.meshgrid(np.arange(-L, L+1), np.arange(-L, L+1), np.arange(-L,␣
↪L+1), indexing='ij')

r = np.sqrt(i**2 + j**2 + k**2)
r[L, L, L] = 1

sign = np.where((i + j + k) % 2 == 0, 1, -1)

potentials = sign / r

total_sum = np.sum(potentials) - potentials[L, L, L]

end_time = timeit.default_timer()
```

```
elapsed_time = end_time - start_time

print(f"The Madelung constant for L = {L} is approximately {total_sum}")
print(f"Time = {elapsed_time}")
```

```
The Madelung constant for L = 325 is approximately -1.7493383281793022
Time = 25.134665900026448
```

6. Exercise 2.10



Exercise 2.10: The semi-empirical mass formula

In nuclear physics, the semi-empirical mass formula is a formula for calculating the approximate nuclear binding energy B of an atomic nucleus with atomic number Z and mass number A:

$$B = a_1 A - a_2 A^{2/3} - a_3 \frac{Z^2}{A^{1/3}} - a_4 \frac{(A-2Z)^2}{A} + \frac{a_5}{A^{1/2}},$$

where, in units of millions of electron volts, the constants are $a_1 = 15.8$, $a_2 = 18.3$, $a_3 = 0.714$, $a_4 = 23.2$, and

$$a_5 = \begin{cases} 0 & \text{if } A \text{ is odd,} \\ 12.0 & \text{if } A \text{ and } Z \text{ are both even,} \\ -12.0 & \text{if } A \text{ is even and } Z \text{ is odd.} \end{cases}$$

a) Write a program that takes as its input the values of A and Z, and prints out the binding energy for the corresponding atom. Use your program to find the binding energy of an atom with $A = 58$ and $Z = 28$. (Hint: The correct answer is around 500 MeV.)

b) Modify your program to print out not the total binding energy B, but the binding energy per nucleon, which is B/A.

c) Now modify your program so that it takes as input just a single value of the atomic number Z and then goes through all values of A from $A = Z$ to $A = 3Z$, to find the one that has the largest binding energy per nucleon. This is the most stable nucleus with the given atomic number. Have your program print out the value of A for this most stable nucleus and the value of the binding energy per nucleon.

d) Modify your program again so that, instead of taking Z as input, it runs through all values of Z from 1 to 100 and prints out the most stable value of A for each one. At what value of Z does the maximum binding energy per nucleon occur? (The true answer, in real life, is $Z = 28$, which is nickel.)

[83]:
```
def calc_a5(A, Z):
    if (A % 2 == 1):
        return 0
    elif (A % 2 == 0 and Z % 2 == 0):
        return 12
    elif (A % 2 == 0 and Z % 2 == 1):
        return -12

def calc_b(A, Z, a5):
    a1 = 15.8
    a2 = 18.3
    a3 = 0.714
    a4 = 23.2
    return a1*A - a2*A**(2/3) - a3*(Z**2/A**(1/3)) - a4*((A - 2*Z)**2/A) + a5/
 ↪A**(1/2)

def most_stable_A(Z):
    max_bepn = -1
    A_max = Z

    for A in range(Z, 3*Z + 1):
        a5 = calc_a5(A, Z)
        B = calc_b(A, Z, a5)
        bepn = B/A

        if bepn > max_bepn:
            max_bepn = bepn
            A_max = A

    return A_max, max_bepn

A = int(input("Enter Mass Number (A):"))
Z = int(input("Enter Atomic Number (Z):"))
a5 = calc_a5(A, Z)

B = calc_b(A, Z, a5)
print(f"Binding Energy (B) = {B}")

bepn = B/A
print(f"Binding Energy Per Nuclean = {bepn}")

Z = int(input("Enter Atomic Number (Z):"))

A_max, max_bepn = most_stable_A(Z)
print(f"The most stable nucleus has A = {A_max}")
print(f"Binding Energy Per Nucleon = {max_bepn}")

for Z in range(1, 101):
    A_max, max_bepn = most_stable_A(Z)
    print(f"For Z = {Z}, the most stable A = {A_max}, with Binding Energy Per␣
 ↪Nucleon = {max_bepn}")
```

```
Enter Mass Number (A): 58
Enter Atomic Number (Z): 28

Binding Energy (B) = 497.5620206224374
Binding Energy Per Nuclean = 8.578655527973059

Enter Atomic Number (Z): 28

The most stable nucleus has A = 62
```

```
Binding Energy Per Nucleon = 8.70245768367189
For Z = 1, the most stable A = 3, with Binding Energy Per Nucleon =
0.36869091831015827
For Z = 2, the most stable A = 4, with Binding Energy Per Nucleon =
5.321930578649441
For Z = 3, the most stable A = 7, with Binding Energy Per Nucleon =
5.280168164356119
For Z = 4, the most stable A = 8, with Binding Energy Per Nucleon =
6.466330085889912
For Z = 5, the most stable A = 11, with Binding Energy Per Nucleon =
6.650123444727665
For Z = 6, the most stable A = 14, with Binding Energy Per Nucleon =
7.200918138809924
For Z = 7, the most stable A = 15, with Binding Energy Per Nucleon =
7.330860591990981
For Z = 8, the most stable A = 18, with Binding Energy Per Nucleon =
7.719275577459026
For Z = 9, the most stable A = 19, with Binding Energy Per Nucleon =
7.73697768275634
For Z = 10, the most stable A = 22, with Binding Energy Per Nucleon =
8.035350864715019
For Z = 11, the most stable A = 25, with Binding Energy Per Nucleon =
8.025554739665797
For Z = 12, the most stable A = 26, with Binding Energy Per Nucleon =
8.241172535624845
For Z = 13, the most stable A = 29, with Binding Energy Per Nucleon =
8.240988355754636
For Z = 14, the most stable A = 30, with Binding Energy Per Nucleon =
8.37916169002579
For Z = 15, the most stable A = 33, with Binding Energy Per Nucleon =
8.38521415855582
For Z = 16, the most stable A = 36, with Binding Energy Per Nucleon =
8.489230168218935
For Z = 17, the most stable A = 37, with Binding Energy Per Nucleon =
8.48201495174352
For Z = 18, the most stable A = 40, with Binding Energy Per Nucleon =
8.573405285254953
For Z = 19, the most stable A = 43, with Binding Energy Per Nucleon =
8.551826855569242
For Z = 20, the most stable A = 44, with Binding Energy Per Nucleon =
8.627152167121634
For Z = 21, the most stable A = 47, with Binding Energy Per Nucleon =
8.610130576802973
For Z = 22, the most stable A = 48, with Binding Energy Per Nucleon =
8.6585154571142
For Z = 23, the most stable A = 51, with Binding Energy Per Nucleon =
8.645234048730842
For Z = 24, the most stable A = 54, with Binding Energy Per Nucleon =
8.687306583887372
For Z = 25, the most stable A = 55, with Binding Energy Per Nucleon =
8.662703971015583
For Z = 26, the most stable A = 58, with Binding Energy Per Nucleon =
8.701432576808987
For Z = 27, the most stable A = 61, with Binding Energy Per Nucleon =
8.678053678353882
For Z = 28, the most stable A = 62, with Binding Energy Per Nucleon =
8.70245768367189
For Z = 29, the most stable A = 65, with Binding Energy Per Nucleon =
8.681907349580422
For Z = 30, the most stable A = 68, with Binding Energy Per Nucleon =
8.701580328486784
For Z = 31, the most stable A = 69, with Binding Energy Per Nucleon =
8.675012598311142
For Z = 32, the most stable A = 72, with Binding Energy Per Nucleon =
8.693433639739787
For Z = 33, the most stable A = 75, with Binding Energy Per Nucleon =
8.668156247337208
For Z = 34, the most stable A = 76, with Binding Energy Per Nucleon =
8.67683411103597
For Z = 35, the most stable A = 79, with Binding Energy Per Nucleon =
8.653727479263061
For Z = 36, the most stable A = 82, with Binding Energy Per Nucleon =
8.66141248935323
For Z = 37, the most stable A = 85, with Binding Energy Per Nucleon =
8.633940444065898
For Z = 38, the most stable A = 86, with Binding Energy Per Nucleon =
8.639441275530453
For Z = 39, the most stable A = 89, with Binding Energy Per Nucleon =
8.613815033672552
For Z = 40, the most stable A = 92, with Binding Energy Per Nucleon =
8.614514461544127
For Z = 41, the most stable A = 93, with Binding Energy Per Nucleon =
8.587741675710747
For Z = 42, the most stable A = 96, with Binding Energy Per Nucleon =
8.588337807352417
For Z = 43, the most stable A = 99, with Binding Energy Per Nucleon =
8.561488033970447
For Z = 44, the most stable A = 102, with Binding Energy Per Nucleon =
8.557428804696526
For Z = 45, the most stable A = 103, with Binding Energy Per Nucleon =
8.531857077904819
For Z = 46, the most stable A = 106, with Binding Energy Per Nucleon =
8.52785864674169
For Z = 47, the most stable A = 109, with Binding Energy Per Nucleon =
8.500490244095234
For Z = 48, the most stable A = 110, with Binding Energy Per Nucleon =
8.4940569666179
For Z = 49, the most stable A = 113, with Binding Energy Per Nucleon =
8.46797678166812
For Z = 50, the most stable A = 116, with Binding Energy Per Nucleon =
8.460713172533021
For Z = 51, the most stable A = 119, with Binding Energy Per Nucleon =
8.433224407343122
For Z = 52, the most stable A = 120, with Binding Energy Per Nucleon =
8.424696665334825
For Z = 53, the most stable A = 123, with Binding Energy Per Nucleon =
8.398332486638505
For Z = 54, the most stable A = 126, with Binding Energy Per Nucleon =
8.38868945344295
For Z = 55, the most stable A = 129, with Binding Energy Per Nucleon =
8.361310553455423
For Z = 56, the most stable A = 130, with Binding Energy Per Nucleon =
8.350819725669146
For Z = 57, the most stable A = 133, with Binding Energy Per Nucleon =
8.32443069811602
For Z = 58, the most stable A = 136, with Binding Energy Per Nucleon =
8.313019639868703
For Z = 59, the most stable A = 139, with Binding Energy Per Nucleon =
8.285884693887583
For Z = 60, the most stable A = 140, with Binding Energy Per Nucleon =
8.273583815729522
For Z = 61, the most stable A = 143, with Binding Energy Per Nucleon =
8.247327458286065
For Z = 62, the most stable A = 146, with Binding Energy Per Nucleon =
8.234582872513133
For Z = 63, the most stable A = 149, with Binding Energy Per Nucleon =
8.207769087134613
For Z = 64, the most stable A = 150, with Binding Energy Per Nucleon =
8.193813966434627
For Z = 65, the most stable A = 153, with Binding Energy Per Nucleon =
8.167786207290451
For Z = 66, the most stable A = 156, with Binding Energy Per Nucleon =
8.154024477520512
For Z = 67, the most stable A = 159, with Binding Energy Per Nucleon =
8.127574530301825
For Z = 68, the most stable A = 162, with Binding Energy Per Nucleon =
8.11278037767185
For Z = 69, the most stable A = 163, with Binding Energy Per Nucleon =
8.086373082934724
For Z = 70, the most stable A = 166, with Binding Energy Per Nucleon =
8.071829365612306
For Z = 71, the most stable A = 169, with Binding Energy Per Nucleon =
8.045764583108769
For Z = 72, the most stable A = 172, with Binding Energy Per Nucleon =
8.030368571593026
For Z = 73, the most stable A = 175, with Binding Energy Per Nucleon =
8.004105311543137
For Z = 74, the most stable A = 176, with Binding Energy Per Nucleon =
7.988369073415251
For Z = 75, the most stable A = 179, with Binding Energy Per Nucleon =
7.962697427019249
For Z = 76, the most stable A = 182, with Binding Energy Per Nucleon =
7.946838055684514
For Z = 77, the most stable A = 185, with Binding Energy Per Nucleon =
7.921016888554693
For Z = 78, the most stable A = 188, with Binding Energy Per Nucleon =
7.904575879934101
For Z = 79, the most stable A = 191, with Binding Energy Per Nucleon =
7.87868400232803
For Z = 80, the most stable A = 192, with Binding Energy Per Nucleon =
7.862438691993173
For Z = 81, the most stable A = 195, with Binding Energy Per Nucleon =
7.837047201045294
For Z = 82, the most stable A = 198, with Binding Energy Per Nucleon =
7.82033857608665
For Z = 83, the most stable A = 201, with Binding Energy Per Nucleon =
7.794899333942829
For Z = 84, the most stable A = 204, with Binding Energy Per Nucleon =
7.777785761025879
For Z = 85, the most stable A = 205, with Binding Energy Per Nucleon =
7.75239433360284
For Z = 86, the most stable A = 208, with Binding Energy Per Nucleon =
7.735485475322138
For Z = 87, the most stable A = 211, with Binding Energy Per Nucleon =
7.710478810144077
For Z = 88, the most stable A = 214, with Binding Energy Per Nucleon =
7.693222175964613
For Z = 89, the most stable A = 217, with Binding Energy Per Nucleon =
7.668228396728228
For Z = 90, the most stable A = 220, with Binding Energy Per Nucleon =
7.65068598823387
For Z = 91, the most stable A = 223, with Binding Energy Per Nucleon =
7.625739835440575
For Z = 92, the most stable A = 224, with Binding Energy Per Nucleon =
7.608214013689897
For Z = 93, the most stable A = 227, with Binding Energy Per Nucleon =
7.583639834526779
For Z = 94, the most stable A = 230, with Binding Energy Per Nucleon =
7.566035830526522
For Z = 95, the most stable A = 233, with Binding Energy Per Nucleon =
7.5415108314640555
For Z = 96, the most stable A = 236, with Binding Energy Per Nucleon =
7.523703637516345
For Z = 97, the most stable A = 239, with Binding Energy Per Nucleon =
7.499251800171257
For Z = 98, the most stable A = 242, with Binding Energy Per Nucleon =
7.481279349508352
For Z = 99, the most stable A = 243, with Binding Energy Per Nucleon =
7.456937323389022
For Z = 100, the most stable A = 246, with Binding Energy Per Nucleon =
7.439122944429214
```

[ ]: