Exercise sheet 1

Exercise 1: Programming in Python

The *Bethe-Weizsäcker formula* is a semi-empirical formula for estimating the nuclear binding energy B of an atomic nucleus:

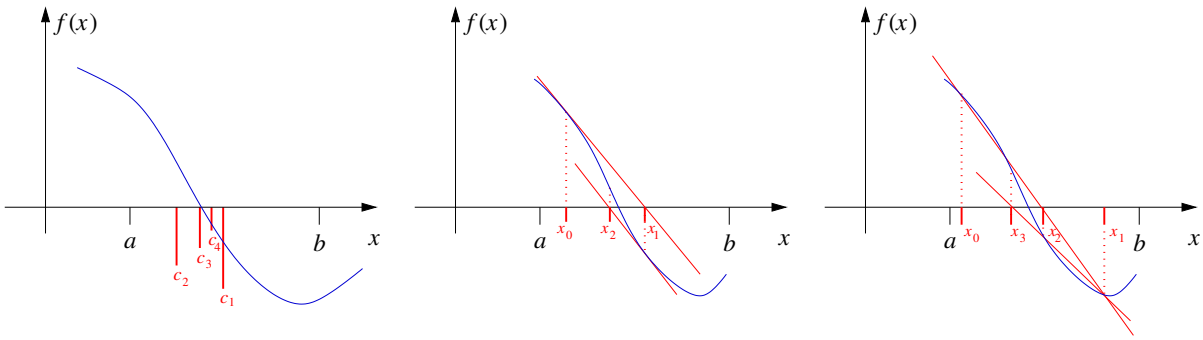
$$B = a_v\, A - a_s\, A^{2/3} - a_c\, Z(Z-1)\, A^{-1/3} - a_a \frac{(A-2\,Z)^2}{A} + a_p\, A^{-1/2}\,.$$

Here A is the atomic mass number (the total number of nucleons), Z is the atomic number (the number of protons), and the as are found to be, in units of MeV,

$$a_v = 15.7\,, \quad a_s = 17.2\,, \quad a_c = 0.714\,, \quad a_a = 23.2\,, \quad a_p = \begin{cases} 0 & A \text{ odd} \\ 11.2 & A \text{ even, } Z \text{ even} \\ -11.2 & A \text{ even, } Z \text{ odd} \end{cases}$$

Write a Python function which determines the most stable isotope for a given Z (i.e. the value of A between $A = Z$ and $A = 3Z$ which maximizes the specific binding energy B/A). Test your function in a program for $Z = 34$ (selenium), which should yield $^{76}_{34}$Se as the most stable isotope with $B/A = 8.91$ MeV.

Exercise 2: Root-finding methods



Left, bisection method; center, Newton's method; right, secant method.

Let $f : \mathbb{R} \to \mathbb{R}$ a continuous function which has a zero on the interval $[a, b]$ (this is guaranteed if the signs of $f(a)$ and $f(b)$ are different, i.e. $f(a)f(b) < 0$). Write three Python functions which find a zero numerically with accuracy ϵ by three different methods:

1. *Bisection*: Assume that $f(a)f(b) < 0$. Divide $[a, b]$ into two halves $[a, c]$ and $[c, b]$, where $c = (a + b)/2$. If $f(a)f(c) < 0$, continue with the left half, $[a, b] \leftarrow [a, c]$. Otherwise, continue with the right half, $[a, b] \leftarrow [c, b]$. Repeat until the length of the interval has shrunk to $< 2\epsilon$ and return its midpoint.

2. *Newton's method*: Assume that f is differentiable and that its derivative is known. Choose an arbitrary $x_0 \in [a, b]$. Calculate $x_{n+1} = x_n - f(x_n)/f'(x_n)$ (initially for $n = 0$) and repeat as long as $|x_{n+1} - x_n| > \epsilon$. Once $|x_{n+1} - x_n| \leq \epsilon$, return x_{n+1}.

3. *Secant method*: Choose two arbitrary x_0 and x_1 in $[a, b]$. Repeatedly calculate $x_{n+1} = \frac{x_{n-1}f(x_n) - x_n f(x_{n-1})}{f(x_n) - f(x_{n-1})}$ (initially for $n = 1$) until $|x_{n+1} - x_n| < \epsilon$. Return x_{n+1}.

Your functions' call signatures should be `bisection(f, a, b, epsilon=1.E-3)` for the bisection method, `newton(f, fprime, x0, epsilon=1.E-3)` for Newton's method, and `secant(f, x0, x1, epsilon=1.E-3)` for the secant method. Test your functions using $f(x) = \sin(x)$ and $[a, b] = [2, 4]$.

Exercise 3: Legendre polynomials

1. The *Legendre polynomials* $P_n(x)$ can be defined by the recurrence relation

$$P_0(x) = 1\,, \qquad P_1(x) = x\,, \qquad P_n(x) = \frac{2n-1}{n} x P_{n-1}(x) - \frac{n-1}{n} P_{n-2}(x)\,.$$

   Write a Python function `P(n, x)` returning $P_n(x)$.

2. Write a program which finds the zeros of $P_5(x)$ by Newton's method with an accuracy $\leq \delta = 10^{-5}$.

   Hints:

   - The n-th Legendre polynomial P_n has n non-degenerate zeros, all lying in the interval $(-1, 1)$. To find the five zeros of P_5 by Newton's method, one needs to choose five suitable starting values x_0. A possible strategy is to first plot P_5 in order to obtain a rough estimate of the zeros' locations, and then use these estimates as starting values.
   - Implement an auxiliary function `dP(n, x)` which calculates $P_n'(x)$ from the recurrence relation

$$P_n'(x) = \frac{nx}{x^2-1} P_n(x) - \frac{n}{x^2-1} P_{n-1}(x)\,.$$

Exercise 4: Numerical data types in Python

Write two versions of a program which calculates the factorial of a given natural number. In the first versions, all variables and constants are of the type `int`, and in the second version, of the type `float`. What does one obtain for 200! with both programs? Explain what you observe.

Exercise 5: Numerical error

1. Write two programs which calculate a numerical approximation of the Euler-Mascheroni constant $\gamma = 0.577\,215\,664\,901\,532\,860\ldots$ using the formulas

$$\gamma = \sum_{k=1}^{\infty} \left(\frac{1}{k} - \ln\left(1 + \frac{1}{k}\right) \right)\,, \quad \gamma = \lim_{n\to\infty} \left(\frac{2^n}{\exp(2^n)} \sum_{m=0}^{\infty} \left(\frac{2^{mn}}{(m+1)!} \sum_{k=0}^{m} \frac{1}{k+1} \right) - n \ln 2 \right)\,.$$

   What kind of error is the dominant one? Which of the two methods is more efficient?

2. (a) Write a program which calculates the roots of the quadratic equation $ax^2+bx+c = 0$ using the standard formula

$$x = \frac{-b \pm \sqrt{\Delta}}{2a}\,, \qquad \Delta = b^2 - 4ac\,.$$

   What do you obtain for $a = c = 0.001$ and $b = 1000$?

(b) Show that the roots can equivalently be written as

$$x = \frac{2c}{-b \mp \sqrt{\Delta}}\,.$$

Modify your program to calculate the roots avec this second formula, and run it with $a = c = 0.001$ et $b = 1000$. What do you obtain? What is the explanation for the difference with the result of (a)? Which one of the formulas is better suited for calculating which one of the roots? Why?

Exercise 6: Analysis of algorithms

1. Show that, for positive constants a, b, c, one has $\Theta(\log(x^a)) \cong \Theta(\log_b x) \cong \Theta(\log(cx)) \cong \Theta(\log(x))$.

2. The following code tests if some given integer n is a prime number. Make sure you understand how exactly the program achieves this, then analyze its complexity: What is the asymptotic growth of $T(n)$ in the worst case?

```
def is_prime(n):
    k = 2
    while k**2 <= n:
        if n % k == 0:
            return False
        k += 1
    return True
```

3. *Reminder*: The matrix product $A \cdot B$ of two $n \times n$ matrices is $(A \cdot B)_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$. Analyze the time complexity of a program which calculates $A \cdot B$ with this formula as a function of n .

4. Consider the root-finding methods of exercise 2. What is the time complexity to find a zero with n-digit precision for the bisection method (assuming that the time it takes to evaluate the function f is negligible)?