# Assignment 4

1. (**3**) Propose a numerically stable way to compute the function $f(x,a) = \sqrt{x+a} - \sqrt{x}$ for positive $x,\ a$.

2. (**5**) Consider numerical evaluation $\mathcal{C} = \tan(10^{100})$ with the help of arbitrary-precision arithmetic module mpmath, which can be called as follows:

```
from mpmath import *
mp.dps = 64 # precision (in decimal places)
mp.pretty = True
+pi
```

   What is the relative condition number of evaluating $\mathcal{C}$ w.r.t the input number $10^{100}$? How many digits do you need to keep at intermediate steps to evaluate $\mathcal{C}$ with 7-digit accuracy?

3. (**5**) Implement the function `solve_quad(b, c)`, receiving coefficients $b$ and $c$ of a quadratic polynomial $x^2+bx+c$, and returning a pair of equation roots. Your function should always return two roots, even for a degenerate case (for example, a call `solve_quad(-2, 1)` should result into `(1, 1)`). Additionally, your function is expected to return complex roots.

   After checking ensuring that your algorithm sort of works, try it on the following 5 tests. Make sure that all of them pass.

```
tests = [{'b': 4.0, 'c': 3.0},
         {'b': 2.0, 'c': 1.0},
         {'b': 0.5, 'c': 4.0},
         {'b': 1e10, 'c': 3.0},
         {'b': -1e10, 'c': 4.0}]
```

4. (**10**) Consider the polynomial

$$w(x) = \Pi_{r=1}^{20}(x-r) = \sum_{i=0}^{20} a_i x^i \tag{1}$$

   and investigate the condition number of roots of this polynomial w.r.t the coefficients $a_i$. To this end, perform the following experiment, using `numpy` root-finding algorithm. Randomly perturb $w(x)$ by replacing the coefficients $a_i \to n_i a_i$, where $n_i$ is drawn from a normal distribution of mean 1 and variance $10^{-10}$. Show the results of 100 such experiments in a single plot, along with the roots of the unperturbed polynomial $w(x)$. Using one of the experiments, estimate the relative and absolute condition number of the problem of finding the roots of $w(x)$ w.r.t. polynomial coefficients.

5. (**10**) Cosider computing the function $f(n,\alpha)$ defined by $f(0,\alpha) = \ln(1+1/\alpha)$ and recurrent relation

$$f(n,\alpha) = \frac{1}{n} - \alpha f(n-1,\alpha). \tag{2}$$

   Compute $f(20, 0.1)$ and $f(20, 10)$ in standard (double) precision. Now, do the same exercise in arbitrary precision arithmetic:

```
from mpmath import mp, mpf
mp.dps = 64 # precision (in decimal places)
f = mp.zeros(1, n)
f[0] = mp.log(1+1/mpf(alpha))
for i in range(1, n):
    f[i] = 1/mpf(i) - mpf(alpha)*f[i-1]
```

   Plot the relative difference between exact and approximate results, in units of machine epsilon `np.finfo(float).eps` for $\alpha = 0.1$ and $\alpha = 10$ as function of $n$. How would you evaluate $f(30, 10)$ without relying on the arbitrary precision arithmetic?

6. (**20**) Consider the least squares problem $Ax \approx b$ at

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1.00001 \\ 1 & 1.00001 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ 0.00001 \\ 4.00001 \end{bmatrix}. \tag{3}$$

- Formally, solution is given by

$$x = (A^T A)^{-1} A^T b, \quad y = Ax. \tag{4}$$

  Using this equation, compute the solution analytically (you may use `sympy`).
- Implement Eq. (4) in `numpy` in single and double precision; compare the results to the analytical one.
- Instead of Eq. (4), implement SVD-based solution to least squares. Which approach is numerically more stable?
- Use `np.linalg.lstsq` to solve the same equation. Which method does this function use?
- What are the four relative condition numbers of this problem, describing sensitivities of $x$ and $y$ to perturbations of $b$ and $A$ ? Give examples of perturbations $\delta b$ and $\delta A$ that approximately attain those condition numbers?

7. (**20***) Consider the infinite matrix $A$ with entries $a_{11} = 1$, $a_{12} = 1/2$, $a_{21} = 1/3$, $a_{13} = 1/4$, $a_{22} = 1/5$, $a_{31} = 1/6$, and so on. Compute $\|A\|_2$ with 10 significant digits.