Exercise 5
Numerical Errors, X11, Makefile
(Hand out: 07.05.2025, hand in: 14.05.2025)

---

1. Task *Minimize the error* (3 P)

a) What is the number of calculation steps N, for which an algorithm with an approximation error of $\epsilon_{\text{appr}} \simeq \frac{1}{N}$ gives the minimum total error, when using single precision ($\epsilon_{\text{m}} \simeq 10^{-7}$)? How large will be the total error? (2 P)

b) What would be the optimum number of calculation steps for the same algorithm with double precision ($\epsilon_{\text{m}} \simeq 10^{-15}$) and which total error would be achieved? (1 P)

2. Task *Calculating a power series* (10 P)

The computation of series for function evaluation is a typical problem that occurs in numerics and often needs some thorough thinking to handle the limitations of finite precision. E.g., the exponential

$$e^{-x} = \sum_{n=0}^{\infty} \frac{(-x)^n}{n!} \approx \sum_{n=0}^{N} \frac{(-x)^n}{n!} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots + (-1)^N \frac{x^N}{N!} \tag{1}$$

When we want to calculate the exponential $\exp(-x)$ for $x = 0.1, 1, 10, 100, 1000$, we have to stop the series evaluation at some Nth term. Moreover, the definition of the sum might instruct us to calculate for the nth term x^n and $n!$ and then divide them. This is a *bad* idea. Why? (2P)

Instead, write a program that implements Eq. (1) in a *good* way to calculate the required x, i.e. use the fact that for the nth term you've already calculated the $(n-1)$th term. Stop summation for the Nth term when

$$\left| \frac{(-x)^N/N!}{\sum_{n=0}^{N}(-x)^n/n!} \right| < \epsilon \tag{2}$$

Where ϵ is the desired precision.

a) Write a C++ program that produces for a given argument x a table of the form
N `sum`(N) $|$`sum`$(N) - \exp(-x)|/\exp(-x)$
where $\exp(x)$ uses the builtin exponential function. Use $\epsilon = 10^{-8}$. (3 P)

b) Check that for small negative x your *good* algorithm converges, and that it converges to the correct answer. (1 P)

c) For increasing ("more negative") $|x|$, show that the *good* algorithm converges, but not to the correct value. Show that for larger $|x|$ not even convergence is given, e.g., test for $\exp(-100)$. (1 P)

d) Compare your results, i.e. repeat the steps above, by using the *bad* algorithm. Also, what happens if ϵ is close to the machine precision? How does your algorithm improve (*better* algorithm) when you avoid *subtractive cancellation* completely? (3 P)

3. Task *Graphical output in an X window with Xgraphics and a simple makefile* (5 P + 3 BP)
For showing animated output of our simulations we will use the X11-based library (it is more like a wrapper) `Xgraphics`[1]. All necessary files can be found here.

a) Compile and run the *first* demo program shown in section 5 of the Xgraphics manual. (3 P)
Try to understand the basic elements of this example.
For compilation and linking you can use the following syntax:

```
g++ -c Xgraphics.c
g++ -o program Xgraphics.o program.cpp -lX11
```

(Note that the library `X11` is given with the option `-lX11`, where the letter after the `-` is a lower case "ℓ".)

b) As a preparation for the upcoming programming tasks:

Write a program that shows an animated point on a circular orbit. (3 extra P)

Hint: For each iteration, the previous position in the xy-plane is overplotted by a point with the background color, then the point is plotted at the new position. Then the loop will pause for some time, e.g., by using `system("sleep 0.02") ;` which calls the Unix command `sleep`.

[1] You can also find `Xgraphics.h`, `Xgraphics.c` and the manual `Xgraphics.pdf` in my *home directory* at `~htodt` on the student's computer cluster.