Problem 5-8: Approximating π by extrapolation

In [Lecture → Section 5.2.3.3] we learned about the approximation of a limit $\lim_{h\to 0}\psi(h)$ based on the evaluation of the function ψ "far away" from 0. In this exercise we will practice the underlying technique of "Lagrangian polynomial extrapolation" for a geometric approximation problem.

Study [Lecture → Section 5.2.3.3] and, in particular [Lecture → Code 5.2.3.19], before your tackle this problem.

In this problem we encounter the situation that a quantity of interest is defined as a limit of a sequence

$$x^* = \lim_{n\to\infty} T(n) \;,$$

where the function $T : \{n_{\min}, n_{\min}+1, \dots\} \mapsto \mathbb{R}$ may be given in procedural form as **double** `T(int n)` only. However, invoking `T(Inf)` will usually result in an error and for very large arguments n the implementation of `T()` may not yield reliable results, *cf.* [Lecture → Ex. 1.5.4.7] and [Lecture → § 5.2.3.16].

The idea of **extrapolation** is, firstly, to compute a few values $T(n_0), T(n_1), \dots, T(n_k)$, $k \in \mathbb{N}$, and to consider them as the values $g(1/n_0), g(1/n_1), \dots, g(1/n_k)$ of a continuous function

$$g : (0, 1/n_{\min}] \mapsto \mathbb{R}, \quad \text{for which, obviously} \quad x^* = \lim_{h\to 0} g(h) \;.$$

Secondly, the function g is approximated by an interpolating polynomial $p_k \in \mathcal{P}_k$ with $p_k(n_j^{-1}) = T(n_j)$, $j = 0, \dots, k$. In many cases we can expect that $p_k(0)$ will provide a good approximation for x^*.

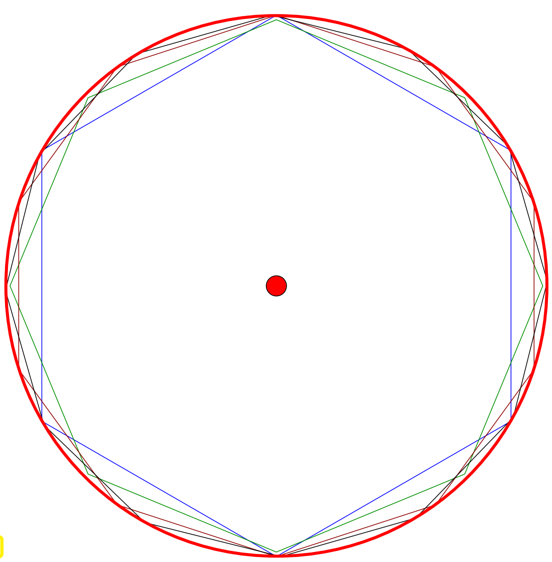

Fig. 36

The unit circle can be approximated by inscribed regular polygons with k edges. Thus, the length of half circumference (i.e., π) can be approximated by the half the length of the perimeters s_k of such polygons. These values $s_k/2$ can be calculated by elementary trigonometry and a closed form formula is

$$\tfrac{1}{2}s_k = k\sin(\pi/k) \;. \tag{5.8.1}$$

(5-8.a) ⚃ (10 min.) Show that for any $L \in \mathbb{N}$ and for $k \to \infty$

$$\tfrac{1}{2}s_k = \pi + \sum_{\ell=1}^{L} c_\ell k^{-2\ell} + O(k^{-2(L+1)}) \quad \text{for some} \quad c_\ell \in \mathbb{R} \quad \text{independent of} \quad L \;. \tag{5.8.2}$$

We say that s_k has an asymptotic expansion in k^{-2}.

Remark. The heuristics behind polynomial extrapolation in this example is that the existence of an asymptotic expansion (5.8.2) suggests that $\frac{1}{2}s_\infty$ can be well approximated by $p(0)$, where p is an interpolating polynomial in k^{-1} or k^{-2}.

HIDDEN HINT 1 for (5-8.a) → 5-8-1-0:s0h1.pdf

SOLUTION for (5-8.a) → 5-8-1-1:s0.pdf ▲

(5-8.b) (45 min.) Implement a C++ function

```
double extrapolate_to_pi(const unsigned int k);
```

that uses the *Aitken-Neville scheme*, see [Lecture → Code 5.2.3.10], to approximate π by extrapolation from the data points $(j^{-1}, \frac{1}{2}s_j)$, for $j = 2, \dots, k$. Use the values $\frac{1}{2}s_j$ as given in (5.8.1).

SOLUTION for (5-8.b) → 5-8-2-0:s1.pdf ▲

(5-8.c) (30 min.) Write a C++ function

```
void plotExtrapolationError(const unsigned int kmax);
```

that generates a *linear-logarithmic* plot of the extrapolation errors

$$\mathrm{err}(k) := |\pi - p_k(0)| \,, \quad k = 2, \dots, k_{\mathsf{max}} \,,$$

versus k and also tabulates the errors. To create the plot use the functions of MATPLOTLIBCPP. Do not forget axis annotations and a meaningful title for the plot.

In `main.cpp`, `plotExtrapolationError(10)` is called for $k_{\mathsf{max}} = 10$.

SOLUTION for (5-8.c) → 5-8-3-0:s2.pdf ▲

(5-8.d) (15 min.) [depends on Sub-problem (5-8.c)]

Which kind of convergence of $\mathrm{err}(k) \to 0$ for $k \to \infty$ can you conclude from the data generated in Sub-problem (5-8.c)? The main types of asymptotic convergence to 0 are introduced in [Lecture → Def. 6.2.2.7].

HIDDEN HINT 1 for (5-8.d) → 5-8-4-0:s3h1.pdf

SOLUTION for (5-8.d) → 5-8-4-1:s3.pdf ▲

(5-8.e) (15 min.) In `main.cpp`, `plotExtrapolationError(30)` is called which plots and tabulates err_k for k up to 30. Describe and explain your observations of the plot found in `cx_out/pi_error_30.png`.

SOLUTION for (5-8.e) → 5-8-5-0:s4.pdf ▲

End Problem 5-8 , 115 min.