

ETH Lectures 401-0663-00L Numerical Methods for Computer Science
401-2673-00L Numerical Methods for CSE

Homework Problems and Projects

Prof. R. Hiptmair, SAM, ETH Zurich

Autumn Term 2021, Version of October 2, 2022
(C) Seminar für Angewandte Mathematik, ETH Zürich

[Link](#) to the current version of this homework collection

0.1 General Information

0.1.1 Weekly Homework Assignments

All problems will be published in this single “.pdf” file. Every week, we publish a list of problem numbers on the lecture Moodle page. These are the current assignments for that week. Details are given on the lecture Moodle page.

0.1.2 Importance of Homework

Homework assignments are **not** mandatory. However, it is **very important** that you constantly exercise with the material you learn. “Solving” the homework assignments one week before the main exam by looking at the solutions will likely result in failure.

We provide hints and solutions from the beginning. It is your responsibility to look at the solutions only if you are stuck with a difficult problem and you tried to find a solution for a sufficient amount of time.

Most homework problems come with an estimate for the time required for its solution. These times are what we expect a bright and well-prepared student to require for that problem during an exam.

0.1.3 Corrections and Grading of Assignments

Homework assignments will not be graded.

However, you may submit your handwritten solutions by handing them to your tutor directly or uploading them via the Moodle upload interface, see the lecture Moodle page for instructions. The assistants will

Problem 5-5: Cardinal basis for trigonometric interpolation

In [Lecture → Section 5.6] we learned about interpolation into the space $\mathcal{P}_{2n}^T$ or trigonometric polynomials from [Lecture → Def. 5.6.1.1]. In this task we investigate the cardinal basis functions for this interpolation operator and will also obtain a result about its stability.

Requires everything from [Lecture → Section 5.6] and involves coding in C++.

The (ordered) real trigonometric basis of $\mathcal{P}_{2n}^T$ is

$$\mathcal{B}_{\text{Re}} := \left\{ C_0 := t \mapsto 1, S_1 := t \mapsto \sin(2\pi t), C_1 := t \mapsto \cos(2\pi t), S_2 := t \mapsto \sin(4\pi t), \right. \\ \left. C_2 := t \mapsto \cos(4\pi t), \dots, S_n := t \mapsto \sin(2n\pi t), C_n := t \mapsto \cos(2n\pi t) \right\}. \quad (5.5.1)$$

Another (ordered) basis of $\mathcal{P}_{2n}^T$ is

$$\mathcal{B}_{\text{exp}} := \{ B_k := t \mapsto \exp(2\pi i k t) : k = -n, \dots, n \}. \quad (5.5.2)$$

(Strictly speaking, we have to take the real part of these functions.) Both bases have $2n + 1$ elements, which agrees with the dimension of $\mathcal{P}_{2n}^T$ [Lecture → Cor. 5.6.1.6].

(5-5.a)  (15 min.) Give the matrix $\mathbf{S} \in \mathbb{C}^{2n+1, 2n+1}$ that effects the transformation of a coefficient representation with respect to $\mathcal{B}_{\text{Re}}$ into a coefficient representation with respect to $\mathcal{B}_{\text{exp}}$.

HINT 1 for (5-5.a): Recall Euler's formula [Lecture → Eq. (5.6.1.2)] expressing trigonometric functions in terms of the complex exponential function. ┘

SOLUTION of (5-5.a):

Let $S_k(t) := \sin(2\pi k t)$, $C_k(t) := \cos(2\pi k t)$ and $B_k(t) := \exp(2\pi i k t)$. Notice $B_0 \equiv C_0 \equiv 1$. Then, by the definition of the trigonometric functions through **Euler's formula**

$$S_k(t) = \frac{1}{2i} B_{-k}(t) - \frac{1}{2i} B_k(t), \\ C_k(t) = \frac{1}{2} B_{-k}(t) + \frac{1}{2} B_k(t).$$

We get the matrix entries by writing the basis elements of $\mathcal{B}_{\text{Re}}$ in terms of the B_k .

$$\begin{aligned} C_0(t) &= B_0(t), \\ S_1(t) &= \frac{1}{2i} (\exp(2\pi i t) - \exp(-2\pi i t)) = \frac{1}{2i} (B_1(t) - B_{-1}(t)), \\ C_1(t) &= \frac{1}{2} (\exp(2\pi i t) + \exp(-2\pi i t)) = \frac{1}{2} (B_1(t) + B_{-1}(t)), \\ &\vdots \\ S_n(t) &= \frac{1}{2i} (\exp(2\pi i n t) - \exp(-2\pi i n t)) = \frac{1}{2i} (B_n(t) - B_{-n}(t)), \\ C_n(t) &= \frac{1}{2} (\exp(2\pi i n t) + \exp(-2\pi i n t)) = \frac{1}{2} (B_n(t) + B_{-n}(t)). \end{aligned}$$

Pay attention to the indices! The basis function C_0 is associated with matrix column 1, S_k , with matrix column $2k$, C_k with matrix column $2k + 1$, $k = 1, \dots, n$. Further B_{k-n} is associated with matrix row

$k+1, k=0, \dots, 2n$. We end up with the basis transformation matrix

$$\mathbf{S} = \begin{bmatrix} 0 & 0 & 0 & \dots & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \frac{1}{2} & \frac{1}{2} & \dots & 0 & 0 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & -\frac{1}{2} & \frac{1}{2} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \in \mathbb{C}^{2n+1, 2n+1}.$$

(5-5.b) (10 min.) The shift operator $S_\tau : C^0(\mathbb{R}) \rightarrow C^0(\mathbb{R})$ acts on a function according to $S_\tau f(t) = f(t - \tau)$, for $\tau \in \mathbb{R}$. If $\mathbf{c} \in \mathbb{C}^{2n+1}$ is the coefficient vector of $p \in \mathcal{P}_{2n}^T$ with respect to $\mathcal{B}_{\text{exp}}$, what is the coefficient vector $\tilde{\mathbf{c}} \in \mathbb{C}^{2n+1}$ of $S_\tau p$?

HINT 1 for (5-5.b): All you need to do is to study the action of S_τ on basis functions.

SOLUTION of (5-5.b):

Consider $S_\tau B_k = \exp(2\pi i k t - 2\pi i k \tau) = B_k \exp(-2\pi i k \tau)$. If $p = \sum_{k=-n}^n c_k B_k(t)$ (expressed in the basis $\mathcal{B}_{\text{exp}}$), then

$$S_\tau p = \sum_{k=-n}^n c_k S_\tau B_k(t) = \sum_{k=-n}^n c_k \exp(-2\pi i k \tau) B_k(t).$$

The new coefficient vector is therefore $\tilde{\mathbf{c}} = [c_k \exp(-2\pi i k \tau)]_{k=-n}^n$.

Now we consider the set of interpolation nodes

$$\mathcal{T}_n := \left\{ t_j := \frac{j + 1/2}{2n + 1} : j = 0, \dots, 2n \right\}. \quad (5.5.3)$$

The **cardinal basis functions** $b_0, \dots, b_{2n} \in \mathcal{P}_{2n}^T$ of $\mathcal{P}_{2n}^T$ with respect to $\mathcal{T}_n$ are defined by

$$b_j(t_k) = \delta_{kj} \quad [\text{Kronecker symbol}], \quad j, k = 0, \dots, 2n \quad (5.5.4)$$

(5-5.c) (30 min.) Use the function `trigpolyvalequid`, see [Lecture → Code 5.6.3.9] to plot the cardinal basis function (implement `plot_basis` in `trigpicardfn.hpp`) $t \mapsto b_0(t)$ in $[0, 1]$ for $n = 5$. To that end evaluate $b_0(t)$ in 1000 equidistant points $\frac{k}{1000}, k = 0, \dots, 999$.

SOLUTION of (5-5.c):

We have to feed unit vectors as y -argument to `trigpolyvalequid` and $M=1000$. Note that the function uses the equidistant points $t_j = \frac{j}{2n+1}$, $j = 0, \dots, 2n$, which does not match the node set $\mathcal{T}_n^{b_0(t)}$ from (5.5.3). Fortunately this can be compensated by reading the returned values as values for evaluation points $x_k = \frac{k}{M} + \frac{1}{4N+2}$ and taking into account 1-periodicity.

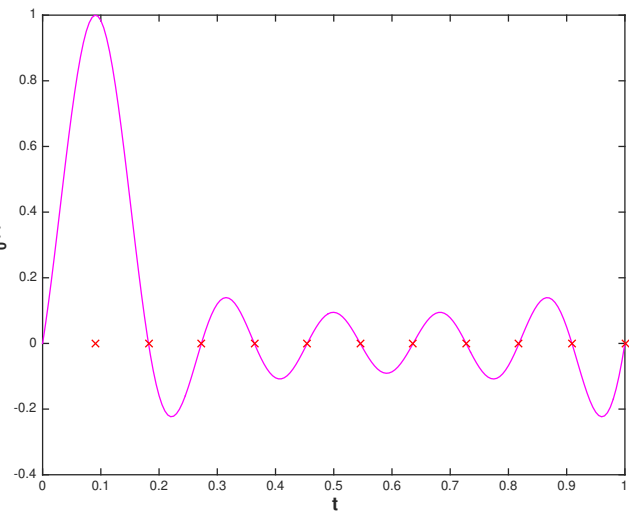


Fig. 32

C++11-code 5.5.5: Plotting of the cardinal basis function b_0 .

```

2 void plot_basis(unsigned int n) {
3     // mesh size
4     constexpr unsigned int M = 1e3;
5     plt::figure();
6
7     // TODO: (5-5.c) use the function trigpolyvalequid from
8     // trigipcardfn.hpp to plot the cardinal basis function  $b_0(t)$  in
9     // of  $t$  for  $n = 5$ 
10    // START
11    // basis vector  $e_1$ 
12    Eigen::ArrayXd e = Eigen::ArrayXd::Zero(2 * n + 1);
13    e(0) = 1;
14    Eigen::VectorXd y;
15    trigpolyvalequid(e, M, y);
16
17    Eigen::ArrayXd t = Eigen::ArrayXd::LinSpaced(M, 0, 1);
18    Eigen::ArrayXd zer = Eigen::ArrayXd::Zero(t.size());
19
20    // Shift function right a bit
21    Eigen::ArrayXd y_shift(M);
22    const unsigned int h = M / (2 * n + 1);
23    y_shift << y.tail(h), y.head(M - h);
24
25    // plotting functions using matplotlib
26    plt::title("b_0(t)");
27    plt::xlabel("t");
28    plt::ylabel("y");
29    plt::plot(t, y_shift, "r", {"label", "b_0(t)"});
30    plt::plot(t, zer, "--");
31    plt::legend();
32    // END
33    plt::savefig("./cx_out/basis_function.png");
34 }

```

Get it on [GitLab](#) (trigipcardfn.hpp).



(5-5.d) ☐ Show that

$$b_{k+1} = S_\delta b_k \quad \text{with} \quad \delta := \frac{1}{2n+1}. \quad (5.5.6)$$

HINT 1 for (5-5.d): Verify (5.5.4) for $S_\delta b_k$ ┘

SOLUTION of (5-5.d):

We have (use periodicity and equidistance of the points):

$$S_\delta b_k(t_j) = b_k(t_j - \frac{1}{2n+1}) = b_k(\frac{(j-1) + 1/2}{2n+1}) = b_k(t_{j-1}) = \delta_{k,j-1} = \delta_{k+1,j} = b_{k+1}(t_j).$$

The claim follows from uniqueness of interpolating trigonometric polynomials. ▲

(5-5.e) ☐ (15 min.) Describe the entries of the “interpolation matrix”, that is, the system matrix of [Lecture → Eq. (5.1.0.23)], for the trigonometric interpolation problem with node set $\mathcal{T}_n$ and with respect to the basis $\mathcal{B}_{\text{exp}}$ of $\mathcal{P}_{2n}^T$.

HINT 1 for (5-5.e): Remember the **Fourier matrix** [Lecture → Eq. (4.2.1.13)]. ┘

HINT 2 for (5-5.e): Examine [Lecture → Eq. (5.6.3.3)]. However, take into account that the interpolation nodes for this problem are slightly different from those used in those formulas. ┘

SOLUTION of (5-5.e):

The trigonometric interpolation problem with respect to the basis $\mathcal{B}_{\text{exp}} = \{B_{-n}, \dots, B_n\}$ from (5.5.2) leads to the linear system of equations

$$\sum_{k=-n}^n c_k B_k(t_j) = y_j, \quad j = 0, \dots, 2n,$$

for given values $y_j \in \mathbb{C}$. The related interpolation matrix is

$$\mathbf{A} := \begin{bmatrix} B_{-n}(t_0) & \dots & B_n(t_0) \\ \vdots & & \vdots \\ B_{-n}(t_{2n}) & \dots & B_n(t_{2n}) \end{bmatrix} \in \mathbb{C}^{2n+1, 2n+1}.$$

The indexing of the exponential basis functions as defined in (5.5.2) runs from $-n, \dots, n$ and has to be shifted to match **C++ indexing** of matrix entries:

$$\begin{aligned} (\mathbf{A})_{j,\ell} &= B_{\ell-n}(t_j) = \exp(2\pi i \frac{(j+1/2)(\ell-n)}{2n+1}) = \exp(2\pi i \frac{j\ell + \frac{1}{2}\ell - jn - \frac{1}{2}n}{2n+1}) \\ &= \underbrace{\exp(2\pi i \frac{j\ell}{2n+1})}_{=(\bar{\mathbf{F}}_{2n+1})_{j,\ell}} \exp(\pi i \frac{\ell}{2n+1}) \exp(-2\pi i \frac{jn}{2n+1}) \exp(-\pi i \frac{1}{2n+1}), \end{aligned}$$

where $\mathbf{F}_{2n+1} \in \mathbb{C}^{2n+1, 2n+1}$ is the **Fourier matrix**. So we find that the interpolation matrix agrees with a Fourier matrix to which some row and column scaling is applied:

$$\mathbf{A} = \exp(-\pi i \frac{n}{2n+1}) \mathbf{D}_1 \bar{\mathbf{F}}_{2n+1} \mathbf{D}_2, \quad (5.5.7)$$

$$\mathbf{D}_1 := \text{diag} \left(\left[e^{-2\pi i \frac{jn}{2n+1}} \right]_{j=0, \dots, 2n} \right), \quad \mathbf{D}_2 := \text{diag} \left(\left[e^{\pi i \frac{\ell}{2n+1}} \right]_{\ell=0, \dots, 2n} \right).$$



(5-5.f) 🕒 (15 min.) [depends on Sub-problem (5-5.e)]

What is the inverse of the interpolation matrix found in the previous problem?

HINT 1 for (5-5.f): Recall [Lecture → Lemma 4.2.1.14].



SOLUTION of (5-5.f):

We exploit the factorization (5.5.7), remember from linear algebra $(\mathbf{XY})^{-1} = \mathbf{Y}^{-1}\mathbf{X}$ for all regular square matrices $\mathbf{X}$, $\mathbf{Y}$, and use the fact that $\exp(i\zeta)^{-1} = \exp(-i\zeta)$ for all $\zeta \in \mathbb{R}$, and get

$$\begin{aligned} \mathbf{A}^{-1} &= e^{\pi i \frac{n}{2n+1}} \mathbf{D}_2^{-1} (\bar{\mathbf{F}}_{2n+1})^{-1} \mathbf{D}_1^{-1} \\ &= e^{\pi i \frac{n}{2n+1}} \text{diag} \left(\left[e^{-\pi i \frac{\ell}{2n+1}} \right]_{\ell=0, \dots, 2n} \right) \frac{1}{2n+1} \mathbf{F}_{2n+1} \text{diag} \left(\left[e^{2\pi i \frac{jn}{2n+1}} \right]_{j=0, \dots, 2n} \right), \end{aligned} \quad (5.5.8)$$

where we also used

Lemma [Lecture → Lemma 4.2.1.14]. Properties of Fourier matrices

The scaled **Fourier-matrix** $\frac{1}{\sqrt{n}} \mathbf{F}_n$ is unitary (→ [Lecture → Def. 6.3.1.2]):
 $\mathbf{F}_n^{-1} = \frac{1}{n} \mathbf{F}_n^H = \frac{1}{n} \bar{\mathbf{F}}_n.$

Based on

$$\mathbf{F}_{2n+1} = \left[\exp(-2\pi i \frac{j\ell}{2n+1}) \right]_{j,\ell=0}^{2n} \in \mathbb{C}^{2n+1, 2n+1},$$

we can recast the formula for $\mathbf{A}^{-1}$ as

$$\begin{aligned} \mathbf{A}^{-1} &= \frac{1}{2n+1} \text{diag} \left(\left[e^{\pi i \frac{n-j}{2n+1}} \right]_{j=0, \dots, 2n} \right) \left[e^{-2\pi i \frac{j\ell}{2n+1}} \right]_{j,\ell=0}^{2n} \text{diag} \left(\left[e^{2\pi i \frac{\ell n}{2n+1}} \right]_{\ell=0, \dots, 2n} \right) \\ &= \frac{1}{2n+1} \left[e^{-2\pi i \frac{\frac{1}{2}(n-j)-j\ell+\ell n}{2n+1}} \right]_{j,\ell=0}^{2n} = \frac{1}{2n+1} \mathbf{A}^H, \end{aligned}$$

by matching with

$$(\mathbf{A})_{j,\ell} = \exp(2\pi i \frac{(j+1/2)(\ell-n)}{2n+1}).$$



(5-5.g) 🕒 (45 min.) Give a closed-form expression for the cardinal basis functions b_k defined in (5.5.4).

HINT 1 for (5-5.g): Use that b_k has the coefficient vector $\mathbf{X}\mathbf{e}_{k+1}$ with respect to $\mathcal{B}_{\text{exp}}$, where $\mathbf{e}_j \in \mathbb{C}^{2n+1}$, $j = 1, \dots, 2n+1$, is the j -th unit vector, $\mathbf{X}$ the matrix found in (5-5.f).



SOLUTION of (5-5.g):

See P. D. MORRIS AND E. W. CHENEY, *Stability properties of trigonometric interpolating operators*, Math. Z., 131 (1973), pp. 153–164.

Let $\mathbf{c}$ be the coefficient vector of b_k w.r.t. $\mathcal{B}_{exp}$. We have

$$\mathbf{c} = \mathbf{A}^{-1} \mathbf{e}_{k+1} = \frac{1}{2n+1} \mathbf{A}^H \mathbf{e}_{k+1} = \frac{1}{2n+1} \left(\exp(-2\pi i(k+1)\frac{j}{2n+1}) \exp(-\pi i \frac{j}{2n+1}) \right)_{j=-n}^n.$$

Expressing b_k with the basis:

$$\begin{aligned} b_k(t) &= \sum_{j=-n}^n c_j B_j(t) = \sum_{j=-n}^n \frac{1}{2n+1} (\exp(2\pi i(k+1)\frac{j}{2n+1}) \exp(-\pi i \frac{j}{2n+1}) \exp(-2\pi i j t)) \\ &= \frac{1}{2n+1} \sum_{j=-n}^n (\exp(2\pi i(k+1/2)\frac{j}{2n+1} - 2\pi i j t)) \end{aligned}$$

Using Euler's formula

$$(n + \frac{1}{2})b_k(t) = \frac{1}{2} + \sum_{j=1}^n \cos(2\pi j(t - t_k)), \quad t \in \mathbb{R}. \quad (5.5.9)$$

This can be simplified further using the geometric sum formula and a few manipulations:

$$\begin{aligned} &\frac{1}{2} + \sum_{j=1}^n \cos(2\pi j(t - t_k)) \\ &= \operatorname{Re}(\frac{1}{2} + \sum_{j=1}^n \exp(2\pi i j(t - t_k))) \\ &= \operatorname{Re}(-\frac{1}{2} + \frac{1 - \exp(2\pi i(n+1)(t - t_k))}{1 - \exp(2\pi i(t - t_k))}) \\ &= \operatorname{Re}(-\frac{1}{2} + \frac{\exp(-\pi i(t - t_k)) - \exp(2\pi i(n+1)(t - t_k)) \exp(-\pi i(t - t_k))}{\exp(-\pi i(t - t_k)) - \exp(\pi i(t - t_k))}) \\ &= \operatorname{Re}(-\frac{1}{2} \frac{\exp(-\pi i(t - t_k)) - \exp(\pi i(t - t_k)) - 2\exp(-\pi i(t - t_k))}{\exp(-\pi i(t - t_k)) - \exp(\pi i(t - t_k))} \\ &\quad - \frac{\exp(2\pi i(n+1)(t - t_k)) \exp(-\pi i(t - t_k))}{\exp(-\pi i(t - t_k)) - \exp(\pi i(t - t_k))}) \\ &= \operatorname{Re}(\frac{1}{2} \frac{\exp(-\pi i(t - t_k)) + \exp(\pi i(t - t_k))}{\exp(-\pi i(t - t_k)) - \exp(\pi i(t - t_k))} - \frac{\exp(2\pi i(n+1)(t - t_k)) \exp(-\pi i(t - t_k))}{\exp(-\pi i(t - t_k)) - \exp(\pi i(t - t_k))}) \\ &= \frac{1}{2} \operatorname{Re}(\frac{\cos(\pi(t - t_k))}{-\sin(\pi(t - t_k))}) - \operatorname{Re}(\frac{\exp(2\pi i(n+1)(t - t_k)) \exp(-\pi i(t - t_k))}{i \sin(\pi(t - t_k))}) \\ &= \frac{\sin(2\pi(n+1)(t - t_k) - \pi(t - t_k))}{\sin(\pi(t - t_k))} \\ &= \frac{\sin(2\pi(n+1/2)(t - t_k))}{\sin(\pi(t - t_k))} \end{aligned}$$

Hence:

$$(n + \frac{1}{2})b_k(t) = \frac{\sin(2\pi(n + \frac{1}{2})t)}{2 \sin(\pi(t - t_k))}, \quad t \in \mathbb{R}. \quad (5.5.10)$$



(5-5.h) 📄 (30 min.) Write a function

```
double trigIpL(std::size_t n);
```

that approximately computes the **Lebesgue constant** $\lambda(n)$, see [Lecture → § 5.2.4.8], for trigonometric interpolation based on the node set $\mathcal{T}_n$. The Lebesgue constant is available through the formula

$$\lambda(n) = \sup_{\mathbf{y} \in \mathbb{C}^{2n+1} \setminus \{0\}} \frac{\|l_n \mathbf{y}\|_{L^\infty([0,1])}}{\|\mathbf{y}\|_\infty} = \sup_{t \in [0,1]} \sum_{k=0}^{2n} |b_k(t)|, \quad (5.5.11)$$

where $l_n : \mathbb{C}^{2n+1} \rightarrow C^0(\mathbb{R})$ is the trigonometric interpolation operator.

Use sampling in 10^4 equidistant points to obtain a good guess for the supremum norm of a function on $[0, 1]$.

Note. Efficiency of the implementation need not be a concern in this numerical experiment.

SOLUTION of (5-5.h):

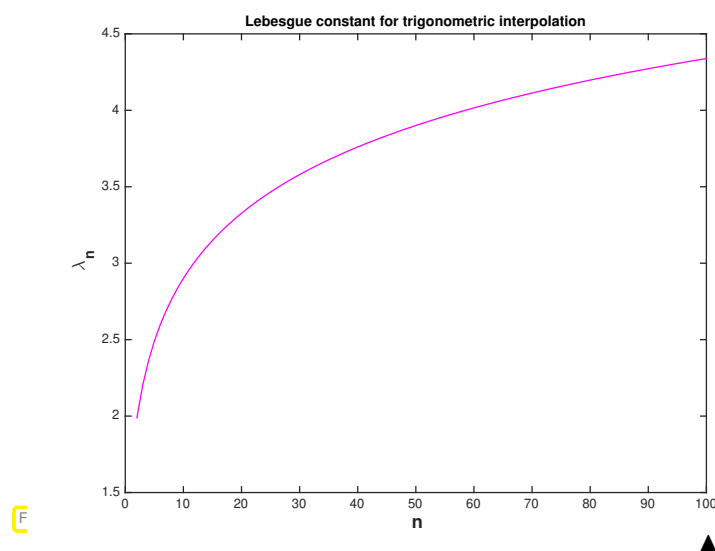
The following code is not optimal. A better, but more complicated implementation could rely on FFT.

C++11-code 5.5.12: Computation of Lebesgue constant λ for trigonometric interpolation.

```
2 double trigIpL(std::size_t n) {
3     double lambda = 0.;
4     // TODO: (5-5.h) write a function that approximately computes the
5     // Lebesgue constant  $\lambda(n)$ 
6     // START
7     // Nodes where to evaluate
8     Eigen::ArrayXd t = Eigen::ArrayXd::LinSpaced(1e4, 0, 1);
9     // Will contain value of function at all nodes
10    Eigen::ArrayXd s = Eigen::ArrayXd::Zero(t.size());
11    // Sum over all basis functions
12    for (unsigned int k = 0; k <= 2 * n; ++k) {
13        // Compute the  $b_k$  using formula (6.5.8)
14        Eigen::ArrayXd b = Eigen::ArrayXd::Zero(t.size());
15        const double tk = (k + 0.5) / (2 * n + 1.);
16        b = sin(M_PI * (2 * n + 1) * (t - tk)) / sin(M_PI * (t - tk));
17
18        s += b.cwiseAbs();
19    }
20    // Find max and rescale
21    lambda = s.maxCoeff() / (2 * n + 1);
22    // END
23    return lambda;
24 }
```

Get it on 🐙 GitLab (trigipcardfn.hpp).

Graph of $n \mapsto \lambda(n)$



(5-5.i) (5 min.) The following table gives the values $\lambda(n)$ for $n = 2^k$, $k = 2, 3, \dots, 8$.

2^k	$\lambda(2^k)$
4	2.7
8	3.07
16	3.46
32	3.89
64	4.32
128	4.75
256	5.18
512	5.62

From these numbers predict the asymptotic behavior of $\lambda(n)$ for $n \rightarrow \infty$. Is it $O(\log n)$, $O(n)$, $O(n \log n)$, or $O(n^2)$?

SOLUTION of (5-5.i):

The asymptotic behavior of $\lambda(n)$ is $O(\log n)$, see E. W. CHENEY AND T. J. RIVLIN, *A note on some Lebesgue constants*, Rocky Mountain J. Math., 6 (1976), pp. 435–439.

End Problem 5-5, 165 min.