Prof. R. Hiptmair
G. Alberti,
F. Leonardi

AS 2015

Numerical Methods for CSE

ETH Zürich
D-MATH

Problem Sheet 8

Problem 1. Natural cubic Splines (core problem)

In [1, Section 3.5.1] we learned about cubic spline interpolation and its variants, the complete, periodic, and natural cubic spline interpolation schemes.

(1a) Given a knot set $\mathcal{T} = \{t_0 < t_1 < \cdots < t_n\}$, which also serves as the set of interpolation nodes, and values y_j, $j = 0, \ldots, n$, write down the linear system of equations that yields the slopes $s'(t_j)$ of the natural cubic spline interpolant s of the data points (t_j, y_j) at the knots.

(1b) Argue why the linear system found in subsubsection (1a) has a unique solution.

HINT: Look up [1, Lemma 1.8.12] and apply its assertion.

(1c) Based on EIGEN devise an *efficient* implementation of a C++ class for the computation of a natural cubic spline interpolant with the following definition:

```
class NatCSI {
public:
    //! \brief Build the cubic spline interpolant with natural boundaries
    //! Setup the data structures you need.
    //! Pre-compute the coefficients of the spline (solve system)
    //! \param[in] t, nodes of the grid (for pairs (t_i, y_i)) (sorted!)
    //! \param[in] y, values y_i at t_i (for pairs (t_i, y_i))
```

```cpp
    NatCSI(const const std::vector<double> & t, const
       const std::vector<double> & y);

    //! \brief Interpolant evaluation at x
    //! \param[in] x, value x where to evaluate the
       spline
    //! \return value of the spline at x
    double operator() (double x) const;

private:
    // TODO: store data for the spline
};
```

HINT: Assume that the input array of knots is sorted and perform binary searches for the evaluation of the interpolant.

Problem 2. Monotonicity preserving interpolation (core problem)

This problem is about monotonicity preserving interpolation. Before starting, you should revise [1, Def. 3.1.15], [1, § 3.3.2] and [1, Section 3.4.2] carefully.

(2a) ⚃ Prove [1, Thm. 3.4.17]:

> If, for fixed node set $\{t_j\}_{j=0}^{n}$, $n \geq 2$, an interpolation scheme $\mathsf{I} : \mathbb{R}^{n+1} \to C^1(I)$ is *linear* as a mapping from data values to continuous functions on the interval covered by the nodes (→ [1, Def. 3.1.15]), and *monotonicity preserving*, then $\mathsf{I}(\mathbf{y})'(t_j) = 0$ for all $\mathbf{y} \in \mathbb{R}^{n+1}$ and $j = 1, \ldots, n-1$.

HINT: Consider a suitable basis $\{\mathbf{s}^{(j)} : j = 0, \ldots, n\}$ of $\mathbb{R}^{n+1}$ that consists of monotonic vectors, namely such that $s_i^{(j)} \leq s_{i+1}^{(j)}$ for every $i = 0, \ldots, n-1$.

HINT: Exploit the phenomenon explained next to [1, Fig. 99].

Problem 3. Local error estimate for cubic Hermite interpolation (core problem)

Consider the cubic Hermite interpolation operator $\mathcal{H}$ of a function defined on an interval $[a, b]$ to the space $\mathcal{P}_3$ polynomials of degree at most 3:

$$\mathcal{H} : C^1([a, b]) \to \mathcal{P}_3$$

defined by:

- $(\mathcal{H}f)(a) = f(a)$;
- $(\mathcal{H}f)(b) = f(b)$;
- $(\mathcal{H}f)'(a) = f'(a)$;
- $(\mathcal{H}f)'(b) = f'(b)$.

Assume $f \in C^4([a,b])$. Show that for every $x \in]a,b[$ there exists $\tau \in [a,b]$ such that

$$(f - \mathcal{H}f)(x) = \frac{1}{24} f^{(4)}(\tau)(x-a)^2(x-b)^2. \tag{18}$$

HINT: Fix $x \in]a,b[$. Use an auxiliary function:

$$\varphi(t) := f(t) - (\mathcal{H}f)(t) - C(t-a)^2(t-b)^2. \tag{19}$$

Find C s.t. $\varphi(x) = 0$.

HINT: Use Rolle's theorem (together with the previous hint and the definition of $\mathcal{H}$) to find a lower bound for the number of zeros of $\varphi^{(k)}$ for $k = 1,2,3,4$.

HINT: Use the fact that $(\mathcal{H}f) \in \mathcal{P}_3$ and for $p(t) := (t-a)^2(t-b)^2, p \in \mathcal{P}_4$.

HINT: Conclude showing that $\exists \tau \in]a,b[, \varphi^{(4)}(\tau) = 0$. Use the definition of φ to find an expression for C.

Problem 4. Adaptive polynomial interpolation

In [1, Section 4.1.3] we have seen that the placement of interpolation nodes is key to a good approximation by a polynomial interpolant. The following *greedy algorithm* attempts to find the location of suitable nodes by its own:

> Given a function $f : [a,b] \mapsto \mathbb{R}$ one starts $\mathcal{T} := \{\frac{1}{2}(b+a)\}$. Based on a fixed finite set $\mathcal{S} \subset [a,b]$ of *sampling points* one augments the set of nodes according to
>
> $$\mathcal{T} = \mathcal{T} \cup \left\{ \operatorname*{argmax}_{t \in \mathcal{S}} |f(t) - \mathsf{I}_{\mathcal{T}}(t)| \right\}, \tag{20}$$
>
> where $\mathsf{I}_{\mathcal{T}}$ is the polynomial interpolation operator for the node set $\mathcal{T}$, until
>
> $$\max_{t \in \mathcal{S}} |f(t) - \mathsf{I}_{\mathcal{T}}(t)| \leq \texttt{tol} \cdot \max_{t \in \mathcal{S}} |f(t)|. \tag{21}$$

(4a) ⚁ Write a MATLAB function

```
function t = adaptivepolyintp(f,a,b,tol,N)
```

that implements the algorithm described above and takes as arguments the function handle `f`, the interval bounds `a`, `b`, the relative tolerance `tol`, and the number `N` of *equidistant* sampling points (in the interval $[a,b]$), that is,

$$\mathcal{S} := \{a + (b-a)\frac{j}{N},\ j = 0,\dots,N\} .$$

HINT: The function `intpolyval` from [1, Code 3.2.28] is provided and may be used (though it may not be the most efficient way to implement the function).

(4b) ⚁ Extend the function from the previous sub-problem so that it reports the quantity

$$\max_{t\in\mathcal{S}} |f(t) - \mathsf{T}_{\mathcal{T}}(t)| \tag{22}$$

for each intermediate set $\mathcal{T}$.

(4c) ⚁ For $f_1(t) := \sin(e^{2t})$ and $f_2(t) = \frac{\sqrt{t}}{1+16t^2}$ plot the quantity from (22) versus the number of interpolation nodes. Choose plotting styles that reveal the qualitative decay of this error as the number of interpolation nodes is increased. Use interval $[a,b] = [0,1]$, `N=1000` sampling points, tolerance `tol = 1e-6`.

Issue date: 05.11.2015

Hand-in: 12.11.2015 (in the boxes in front of HG G 53/54).

Version compiled on: November 12, 2015 (v. 1.0).