Prof. Ch. Schwab
B. Fitzpatrick
J. Zech

Spring Term 2016

Numerische Mathematik I

ETH Zürich
D-MATH

Homework Problem Sheet 6

Introduction. This problem sheet is devoted to polynomial interpolation.

Problem 6.1 Approximation of the First Derivative by Interpolation

A way to compute an approximation of the derivative of a smooth function which is accessible through point values only is to interpolate the function first and then compute the derivative of the interpolant.

(6.1a) Let the polynomial $p \in \mathbb{P}_2$ interpolate the function $f \in C^4(\mathbb{R})$ in the data points 0, h and $2h$. Use $p'(0)$ as an approximation of $f'(0)$ by writing it as

$$p'(0) = \frac{1}{2h}(uf(0) + vf(h) + wf(2h)),$$

for some $u,\ v,\ w \in \mathbb{R}$. Express u, v, and w in terms of $f(0)$, $f(h)$, and $f(2h)$.

Solution: We determine the coefficients of the interpolating polynomial $p_2(x) = a_0 + a_1x + a_2x^2$ as the solution of the system

$$\begin{pmatrix}1 & 0 & 0\\ 1 & h & h^2\\ 1 & 2h & 4h^2\end{pmatrix}\begin{pmatrix}a_0\\ a_1\\ a_2\end{pmatrix} = \begin{pmatrix}f(0)\\ f(h)\\ f(2h)\end{pmatrix},$$

(6.1b) By considering the *Taylor expansion* of f, show that the asymptotic error for $h \to 0$ is given by

$$f'(0) - p'(0) = cf^{(3)}(0)h^2 + \mathcal{O}(h^3)$$

for some constant $c \in \mathbb{R}$. Calculate c explicitly.

Solution: As $f \in C^4(\mathbb{R})$, we can write a Taylor expansion of f up to order four and plug in h and $2h$ to get

$$f(h) = f(0) + hf'(0) + \frac{1}{2}h^2f''(0) + \frac{1}{6}h^3f^{(3)}(0) + \mathcal{O}(h^4),$$

Problem 6.2 Data Compression by Interpolation

Some data sets have a lot of internal structure, which enables compression, that is, a good approximate description with a significantly reduced amount of information. In this problem we will learn about *algorithms* that use interpolation for data compression. This problem asks for substantial implementation in MATLAB.

Assume that the following evaluation function is available

```
v = ipeval(x,y,t),
```

which takes as arguments the data points (x_i, y_i), with $x_{i-1} < x_i$ for $i = 0, \ldots, n-1$ (stored in the row vectors `x` and `y`), and the (sorted) evaluation points $t_1, \ldots, t_N$, $x_0 \leq t_i \leq x_n$ and $t_{i-1} \leq t_i$ (passed as row vector `t`) and returns a row vector $(v_1, \ldots, v_N)$ of values of an interpolating function at $t_1, \ldots, t_N$. For the case of polynomial interpolation such function is `ipolevalbarycentric` and is available on the course webpage as `ipolevalbarycentric.m`.

The following code implements a MATLAB function that effects a transformation of data values based on the interpolation scheme implemented in `ipeval`. To understand how to pass functions as functions arguments, please familiarize yourself with the concept of **function handles** in MATLAB.

Listing 6.1: Implementation of `iprectrf`

```
function y = iprectrf(x,y,ipeval)
% Data transformation by recursive interpolation.
% The function handle ipeval is an evaluation routine
% for an interpolation scheme.

n = length(x);
if (n ~= length(y)), error('length mismatch'); end
if (n > 2)
  % Odd number of data points needed for binary decimation
  if (mod(n,2) ~= 1), error('odd number of points required');
      end
  y(2:2:n-1) = y(2:2:n-1) -
     ipeval(x(1:2:n),y(1:2:n),x(2:2:n-1));
  y(1:2:n) = iprectrf(x(1:2:n),y(1:2:n),ipeval);
end
```

(6.2a) What is the asymptotic complexity of `iprectrf` in terms of the number of data points $n \to \infty$ when a handle to the routine `intpolyval` is passed?

HINT: Code in Listing 6.1 is available on the course webpage as `iprectrf.m`.

First determine the asymptotic computational effort required for `ipolevalbarycentric`.

Solution:

(6.2b) In [NMI, Sect. 3.1] of the lecture notes, linear interpolation of data points was introduced as a simple example for an interpolation scheme: the data points are just connected by straight lines.

Implement a MATLAB function

`p = pwlinintp(x,y,t)`

which is the version of `ipeval` for piecewise linear interpolation. Answer the previous subproblem, if `iprectrf` is called with a handle to this routine.

HINT: You should make use of the knowledge that the nodes and evaluation points are sorted.

(6.2c) Implement a MATLAB function `y = ipinvtrf(x,y,ipeval)` for the inversion of the recursive transformation of data, such that the following expression evaluates to `true`

`y = ipinvtrf(x,iprectrf(x,y,ipeval),ipeval).`

(6.2d) Test your implementation for the inversion of the recursive transformation of data in subproblem (6.2c) when `ipeval` is given by

(i) `ipolevalbarycentric`: polynomial interpolation;

(ii) `pwlinintp`: piecewise linear interpolation;

(iii) MATLAB routine `spline`: cubic spline interpolation (study the documentation of the MATLAB function `spline`).

Use the data `x = cos((2*(n:-1:1)-1)*pi/(2*n))` and `y = rand(1,n)` for $n = 2^L + 1$ and $L = 10$.

(6.2e) The following MATLAB function is purported to provide a compression of the data by recursive spline interpolation. The input is a set of $2^L + 1$ data points with ordered abscissas. The argument `ratio` specifies the compression ratio. The function returns a **structure** containing the relevant coefficients after compression.

Listing 6.2: Implementation of `ipcompress`

```
function cd = ipcompress(x,y,ratio)
% Data compression by recursive spline interpolation

n = length(y);
% transform data values by recursive interpolation
y = iprectrf(x,y,@spline);
% The smallest (in modulus) coefficients will be dropped
d = min(floor((n-2)*ratio),n-2);
z = sort(abs(y(2:end-1)));
cd.idx = [1,(find(abs(y(2:end-1)) > z(d+1)) + 1),n];
cd.values = y(cd.idx);
cd.length = n;
```

Explain, why the routine perform a data *compression*.

HINT: The code in Listing 6.2 is available on the course webpage as `ipcompress.m`.

Solution:

(6.2f) Write a MATLAB function

```
function y = ipuncompress(x,cd)
```

that approximately restores the data compressed by `ipcompress`. Here `x` has to be the same vector passed to `ipcompress` and `cd` is a data structure created by `ipcompress`.

HINT: Use the inversion of recursive spline interpolation.

(6.2g) Plot the data (x, y) given by `x=0:1/(n-1):1`, $y = \cos(\pi e^{3x})$ for $L = 8$ and $n = 2^L + 1$, and the data resulting from 75%, 85%, and 95% compression via `ipcompress`.

(6.2h) Let $\widetilde{\mathbf{y}}(r)$ denote the data vector resulting from recursive spline interpolation compression with compression ratio $0 \leq r \leq 1$ of the data vector $\mathbf{y}$ (with respect to the abscissa points x_i). For the data used in the previous subproblem create a semi-logarithmic plot of $\|\widetilde{\mathbf{y}}(r) - \mathbf{y}\|_2$ versus r. What empiric dependence of $\|\widetilde{\mathbf{y}}(r) - \mathbf{y}\|_2$ on r can you observe?

Problem 6.3 A Cubic Spline

Determine a cubic spline $f : [1, 4] \to \mathbb{R}$ that interpolates a function with values $(y_1, y_2, y_3) = (1, 2, 3)$ at points $(x_1, x_2, x_3) = (1, 2, 4)$ such that

(i) $f(x_i) = y_i$ for $i = 1, 2, 3$,

(ii) $f|_{[x_i, x_{i+1}]}$ are polynomials of degree three for $i = 1, 2$,

(iii) f, f' and f'' are continuous at x_2,

(iv) $f'(x_1) = 0 = f'(x_3)$.

To do this, construct a system of linear equations and solve it in MATLAB. Plot your solution and compare it to the MATLAB function `spline`. The MATLAB command `hold` may be useful to overlay different parts of the plot.

Solution: We set

$$f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i, \quad \text{for } i = 1, 2,$$

where $f_1 = f|_{[x_1, x_2]}$ and $f_2 = f|_{[x_2, x_3]}$.

Listing 6.3: Testcalls for Problem 6.3

```
cubic_spline
```

Listing 6.4: Output for Testcalls for Problem 6.3

```
>> test_call

```

```
f1 =

   -0.7500    4.0000   -5.7500    3.5000

f2 =

    0.0625   -0.8750    4.0000   -3.0000
```

Problem 6.4 Quadratic Splines

Beyond the cubic splines presented in class, splines of general polynomial degree can be defined. One member of the resulting family of piecewise polynomial functions are the *quadratic splines*, which we consider on the interval $[0,1]$ in this problem.

Definition. Given a knot set $\mathcal{N} := \{0 =: x_0 < x_1 < \ldots < x_{n-1} < x_n := 1\}$ we define the space of quadratic splines over $\mathcal{N}$ as

$$\mathcal{S}_{2,\mathcal{N}} := \{s \in C^1([0,1]) : \; s_j = s|_{[x_{j-1},x_j]} \in \mathbb{P}_2 \;\; \forall j = 1,\ldots,n\} \, .$$

In particular, we deal with quadratic spline interpolation of periodic functions: Consider a 1-periodic function $f : \mathbb{R} \to \mathbb{R}$, that is, $f(t+1) = f(t)$ for all $t \in \mathbb{R}$, and a set of nodes

$$\mathcal{N} := \{0 = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = 1\} \subset [0,1] \, .$$

We want to approximate f using a *1-periodic* quadratic spline function $s \in \mathcal{S}_{2,\mathcal{N}}$, which interpolates f in the *midpoints* of the intervals $[x_{j-1}, x_j]$ for $j = 0, \ldots, n$.

Similarly to what was done in Section 3.9 of the lecture notes for cubic splines, we locally parameterize a quadratic spline function $s \in \mathcal{S}_{2,\mathcal{N}}$ as follows. For all $j = 1, \ldots, n$,

$$s|_{[x_{j-1},x_j]}(x) = d_j \, \tau^2 + c_j \, 4\,\tau(1-\tau) + d_{j-1} \, (1-\tau)^2 \, , \quad \tau := \frac{x - x_{j-1}}{x_j - x_{j-1}} \, , \tag{6.4.1}$$

with $c_j, d_k \in \mathbb{R}$, $j = 1, \ldots, n, \; k = 0, \ldots, n$.

(6.4a) How are the coefficients $\{d_j\}_{j=0}^n$ related to particular point values of s?

Solution:

(6.4b) What is the dimension of the subspace of *1-periodic* spline functions in $\mathcal{S}_{2,\mathcal{N}}$?

HINT: Rely on a heuristic counting argument based subtracting the number of linear constraints implied by the continuity requirements from the dimension of the space of $\mathcal{N}$-piecewise quadratic polynomials.

Solution:

(6.4c) What kind of continuity is already guaranteed by the use of the representation (6.4.1)?

Solution:

(6.4d) Derive a linear system of equations (system matrix and right hand side) whose solution provides the coefficients c_j and d_j in the local representation (6.4.1) of a *1-periodic quadratic spline* s that satisfies the interpolation conditions

$$s(\tfrac{1}{2}(x_{j-1}+x_j)) = y_j \ , j = 1, \dots, n \ , \tag{6.4.2}$$

where the values y_j, $j = 1, \dots, n$ are given.

HINT: We know $\mathcal{S}_{2,\mathcal{N}} \subset C^1([0,1])$, which provides linear constraints at the knots x_j, $j = 1, \dots, n$. The constraints attached to x_n are due to periodicity.

Solution: We can plug $x = \frac{1}{2}(x_j + x_{j-1})$ into (6.4.1) and set the values equal to y_j. We obtain $\tau = 1/2$ and the following conditions

$$\frac{1}{4}d_j + c_j + \frac{1}{4}d_{j-1} = y_j, \quad j = 1, ..., n. \tag{6.4.3}$$

We obtain conditions on d_j by matching the derivatives at the interfaces.

Simplifying for d_j we obtain:

$$d_j = \frac{2\frac{c_j}{\Delta_j} + 2\frac{c_{j+1}}{\Delta_{j+1}}}{\frac{1}{\Delta_j} + \frac{1}{\Delta_{j+1}}} = 2\frac{c_j\Delta_{j+1} + c_{j+1}\Delta_j}{\Delta_j + \Delta_{j+1}} = 2\frac{c_j(x_{j+1}-x_j) + c_{j+1}(x_j - x_{j-1})}{x_{j+1} - x_{j-1}}, \ j = 1, \dots, n.$$

Plugging this expression into (6.4.3), we get the following system of equations, for $j = 1, \dots, n$, We collect the coefficients and finally we obtain, for $j = 1, \dots, n$,

(6.4e) Implement an efficient MATLAB function

```
s = quadspline(x,y,t)
```

which takes as input a (sorted) knot vector x (of length $n-1$, because $x_0 = 0$ and $x_n = 1$ will be taken for granted), a n-vector y containing the values y_j of the 1-periodic quadratic spline at the midpoints $\frac{1}{2}(x_{j-1}+x_j)$, $j = 1, \dots, n$, and a *sorted* N-vector t of evaluation points in $[0,1]$.

The function returns the values of the interpolating quadratic spline s at the evaluation points t.

HINT: You should aim for an efficient implementation making use of the fact that the vectors x and t are sorted.

(6.4f) Test your code from subproblem (6.4e) by interpolating the function $f(x) := \mathrm{e}^{\sin(2\pi x)}$ in the interpolation nodes $= \left\{\frac{1}{20} + j/10\right\}_{j=0}^{10}$ with a 1-periodic quadratic splines over the knot set $\mathcal{N} = \{\frac{j}{10}\}_{j=0}^{10}$. Plot the function f and the interpolating quadratic spline s on the interval $[0,1]$ based on the evaluation of both functions in the points $t_j := \frac{j}{200}$, $j = 0, \dots, 200$.

(6.4g) As in subproblem (6.4f) we consider the interpolation of the function $f(x) := \mathrm{e}^{\sin(2\pi x)}$ in the interpolation nodes $= \left\{\frac{1}{2n} + j/n\right\}_{j=0}^{n}$, $n \in \mathbb{N}$, with a 1-periodic quadratic splines s_n over the knot set $\mathcal{N} = \{\frac{j}{n}\}_{j=0}^{n}$. This time we are interested in the behavior of maximum norm of the interpolation error

$$\|f - s_n\|_{\infty,[0,1]} = \sup_{x\in[0,1]} |f(x) - s_n(x)| \,,$$

as a function of n and its asymptotic trend for $n \to \infty$.

Implement a MATLAB script that computes the interpolation error in the maximum norm for $n = 2^2, \ldots, 2^{11}$. Create a loglog plot of the error versus the number n of knot intervals. Describe qualitatively and quantitatively the convergence $s_n \to f$ in the maximum norm for $n \to \infty$.

HINT: As explained in class, the maximum norm of the interpolation error can only be computed approximately. To obtain an approximation evaluate the difference $|f(t) - s_n(t)|$ at $N \gg n$ equidistant evaluation points in $[0, 1]$ and then take the maximum. You may choose $N = 10.000$.

Published on April 8, 2016.

To be submitted on April 19, 2016.

MATLAB: Submit all file in the online system. Include the files that generate the plots. Label all your plots. Include commands to run your functions. Comment on your results.

References

[NMI] Lecture Notes for the course “Numerische Mathematik I”.

Last modified on April 11, 2016