R. Hiptmair
S. Hosseini
C. Pagliantini

Spring Term 2014

Numerical Analysis I

ETH Zürich
D-MATH

Homework Problem Sheet 6

Introduction. This problem sheet is devoted to polynomial interpolation.

Problem 6.1 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` (see slides of the course NMI_3) 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));
```

```
12    y(1:2:n) = iprectrf(x(1:2:n),y(1:2:n),ipeval);
13  end
```

(6.1a) 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`.

(6.1b) 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.

(6.1c) 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).
```

(6.1d) Test your implementation for the inversion of the recursive transformation of data in subproblem (6.1c) 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 $L = 10$.

(6.1e) 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`

```
1  function cd = ipcompress(x,y,ratio)
2  % Data compression by recursive spline interpolation
3
```

```
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`.

(6.1f) 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.

(6.1g) Plot the data (x, y) given by `x=0:1/(n-1):1`, $y = \cos(\pi \mathrm{e}^{3x})$ for $L = 8$ and $n = 2^L + 1$, and the data resulting from 75%, 85%, and 95% compression via `ipcompress`.

(6.1h) 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 6.2 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.

(6.2a) 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 $f(2h)$.

(6.2b) 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.

Problem 6.3 Approximation of the Second Derivative by Interpolation

Often functions $f : I \subset \mathbb{R} \to \mathbb{R}$ that are known to be smooth are available only in procedural form

```
function y = f(x),
```

that is we can only sample them at a finite number of points. The task is to obtain information about derivatives of f from point evaluations. One strategy is to interpolate f and then use derivatives of the interpolant as approximations of derivatives of f. In this problem we see an specimen of the approach based on polynomial interpolation to obtain point values of the second derivative.

Let I be the open interval $(-1, 1) \subset \mathbb{R}$, $f \in C^2(I)$ and $x_0 \in I, h > 0$ such that $x_0 \pm h \in I$. By considering the interpolating polynomial $p_h \in \mathbb{P}_2$ of f for the nodes $x_0 - h$, x_0, and $x_0 + h$, we use $p_h''(x_0)$ as an approximation of $f''(x_0)$.

(6.3a) Give a formula for $p_h''(x_0)$ in terms of $f(x_0 - h)$, $f(x_0)$, and $f(x_0 + h)$.

(6.3b) Based on the *Taylor expansion* of f around x_0, show that the error for $h \to 0$ is asymptotically given by

$$f''(x_0) - p_h''(x_0) = o(1) \,. \tag{6.3.1}$$

HINT: $g(x) = o(1)$ for $x \to 0$ means that $\lim_{x \to 0} g(x) = 0$.

(6.3c) Show that if $f \in C^3(I)$, then the $o(1)$ in (6.3.1) can be replaced with $\mathcal{O}(h)$.

Problem 6.4 Interpolation of singular functions

Obviously, the position of the interpolation nodes has decisive impact on the quality of the interpolant of a function. If special properties of the function are known a priori, we can make an appropriate choice of interpolation nodes. A particular example is given in this problem.

(6.4a) Implement a MATLAB function

```
e = fiterr(a,b,x)
```

which interpolates the function $f \in C^0([0, 1])$, $f(t) = t^\beta$, $\beta = \frac{5}{3}$ in the data points $(x_i, f(x_i))$ for $i = 0, \ldots, n$, $0 \leqslant a \leqslant x_0 < \ldots < x_n \leqslant b \leqslant 1$ and returns the $\|\cdot\|_\infty$-norm of the interpolation error $E_n[f] = f - I^{(n)}[f]$ on the interval $[a, b]$. Test your implementation with the code given in Listing 6.3. You should obtain the value $e \approx 0.0113$.

Listing 6.3: Test call for implementation of subproblem (6.4a)

```
% Test call for fiterr
a = 1/4; b = 1;
x = linspace(1/3,1/2,4);
e = fiterr(a, b, x)
```

HINT: You may use the polynomial interpolation function `ipolevalbarycentric` provided on the course webpage. Do not use the MATLAB routines `polyval` and `polyfit`, because they rely on the monomial basis whose inherent instability was discussed in class.

(6.4b) Let us consider the interpolation problem for $f(t) = t^\beta$, $\beta = \frac{5}{3}$, in $[a,b] = [0,1]$ in the $m+2$ data points $\{x_k^{(m)}\}_{k=0}^{m+1} \subset [a,b]$ defined as

(i) *equidistant* data points $x_k^{(m)} = a + \frac{k}{m+1}(b-a)$ for $k = 0, \ldots, m+1$;

(ii) data points $x_k^{(m)} = 2^{-k}$ for $k = 0, \ldots, m+1$.

Write a MATLAB routine that plots the interpolation error in a loglog plot with respect to $N = m+2$ with $m = 0, \ldots, 16$, for the two sets of data points. Explain the behavior of the error.

(6.4c) We consider the following *piecewise polynomial* interpolation scheme that takes into account the singular behavior of $f(t) = t^\beta$ in $t = 0$. We define for $m \in \mathbb{N}$ and $k = 0, \ldots, m$

- $b_k = 2^{-k}$, $a_k = \frac{1}{2}b_k$, $J_k := (a_k, b_k]$ and $x_\ell^{(k)} := a_k + \frac{\ell}{n}(b_k - a_k)$, $\ell = 0, \ldots, m-k+1$ ($m-k+2$ equidistant interpolation nodes in J_k),
- $p_{m-k+1} \in \mathbb{P}_n$ polynomial interpolant of f in the nodes $x_\ell^{(k)}$, $\ell = 0, \ldots, m-k+1$,
- $p(t) := p_k(t)$, if $t \in J_k$, and $p(t) = 0$, if $t \in [0,1) \setminus \bigcup_{j=0}^m J_j$.

Compute and plot as in subproblem (6.4b) the error $\left\|p^{(m)} - f\right\|_{\infty,[0,1]}$ for $f(t) = x^\beta$, $\beta = \frac{5}{3}$, $m = 0, \ldots, 16$, as a function of $\sqrt{N}$, where N is the number of evaluations of f used for the interpolation. This time you should be able to see the a clear pattern in a `semilogy` plot.

Published on March 31, 2014.

To be submitted on April 8/9, 2014.

MATLAB: Submit all files 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.

Written exercises: Hand-in the solutions during the exercise class or in the labeled boxes in HG G 53.x.

References

[NMI] Lecture Slides for the course "Numerical Analysis I".

Last modified on April 8, 2014