

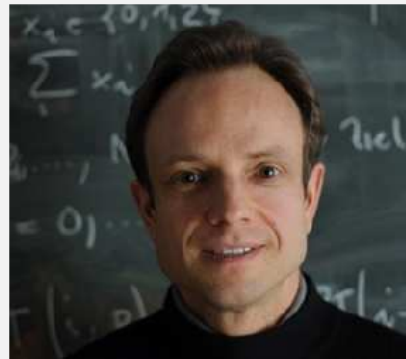
Course Video

Section 5.2.3.3: Extrapolation to Zero

Prof. R. Hiptmair, SAM, ETH Zurich

Date: October 27, 2020

(C) Seminar für Angewandte Mathematik, ETH Zürich



Prerequisites.

- Taylor expansion from calculus

Dependencies. Section 5.2.3, Ex. 1.5.4.7

Duration: 23 minutes



Note: Possible minor mismatch of video and tablet notes!

[Corrections and updates can be incorporated into tablet notes only]

V. Data Interpolation and Data Fitting in 1D

5.2. Global Polynomial Interpolation

5.2.3. Polynomial Interpolation: Algorithms

5.2.3.3. Extrapolation to Zero

Compute $\psi(0)$ for "nice" function $\psi: [-1, 1] \rightarrow \mathbb{R}$
 $\hookrightarrow$ available as `double Psi(double h)`
 $\downarrow$
 not reliable for $|h| \ll 1$

Example 5.2.3.16 (Numerical differentiation through extrapolation)

$$(f \in C^2) \quad \psi(h) := \frac{f(x+h) - f(x-h)}{2h} \stackrel{!}{=} \text{cancellation for } h \approx 0$$

$$\hookrightarrow \psi(0) = f'(x)$$

②

Example:

$f(x) = \arctan(x)$

h	Relative error
2^{-1}	0.20786640808609
2^{-6}	0.00773341103991
2^{-11}	0.00024299312415
2^{-16}	0.00000759482296
2^{-21}	0.00000023712637
2^{-26}	0.00000001020730
2^{-31}	0.00000005960464
2^{-36}	0.00000679016113

$f(x) = \sqrt{x}$

h	Relative error
2^{-1}	0.09340033543136
2^{-6}	0.00352613693103
2^{-11}	0.00011094838842
2^{-16}	0.00000346787667
2^{-21}	0.00000010812198
2^{-26}	0.00000001923506
2^{-31}	0.00000001202188
2^{-36}	0.00000198842224

$f(x) = \exp(x)$

h	Relative error
2^{-1}	0.29744254140026
2^{-6}	0.00785334954789
2^{-11}	0.00024418036620
2^{-16}	0.00000762943394
2^{-21}	0.00000023835113
2^{-26}	0.00000000429331
2^{-31}	0.00000012467100
2^{-36}	0.00000495453865

Taylor expansion, $f \in C^{L+1}$

$$\psi(h) := \frac{f(x+h) - f(x-h)}{2h} = f'(x) + \underbrace{\sum_{\ell=1}^L c_\ell h^\ell}_{\text{a polynomial } p \text{ in } h: p(0) = \psi(0)} + \left[O(h^{L+1}) \right]$$

Small remainder for $|h| \ll 1$

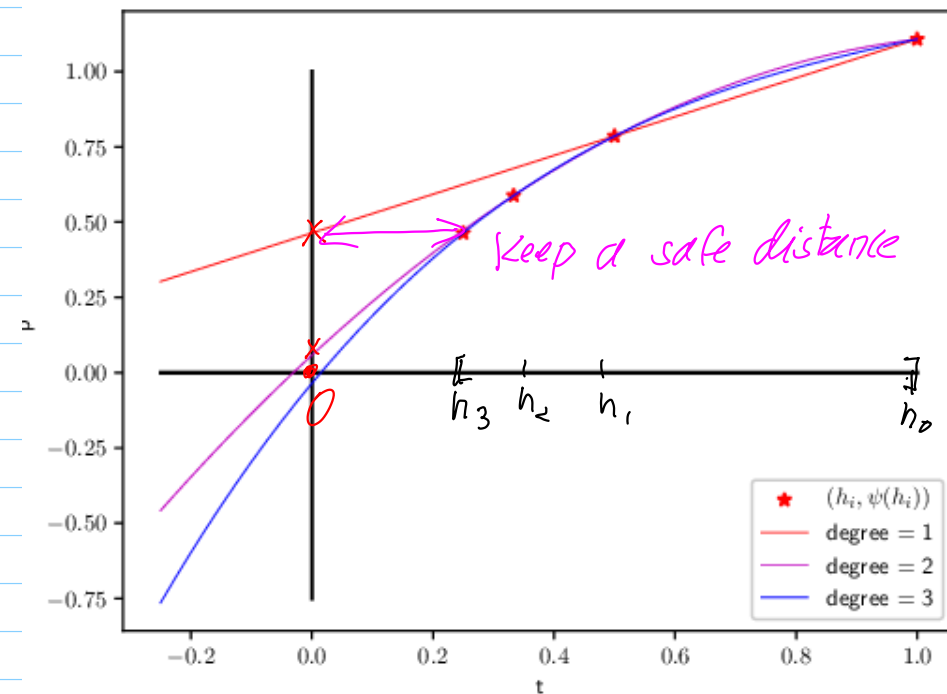
Idea: computing inaccessible limit by extrapolation to zero



- ① Pick $h_0, \dots, h_n$ for which ψ can be evaluated "safely".
 - ② evaluation of $\psi(h_i)$ for different $h_i, i = 0, \dots, n, |t_i| > 0$.
 - ③ $\psi(0) \approx p(0)$ with interpolating polynomial $p \in \mathcal{P}_n, p(h_i) = \psi(h_i)$.
- $\} (h_i, \psi(h_i))$

"Extrapolation"

0 & interval covered by nodes

Algorithm: Compute $p(0)$ by Aitken-Neville

Recall: A.N. "update-friendly", for single evaluation

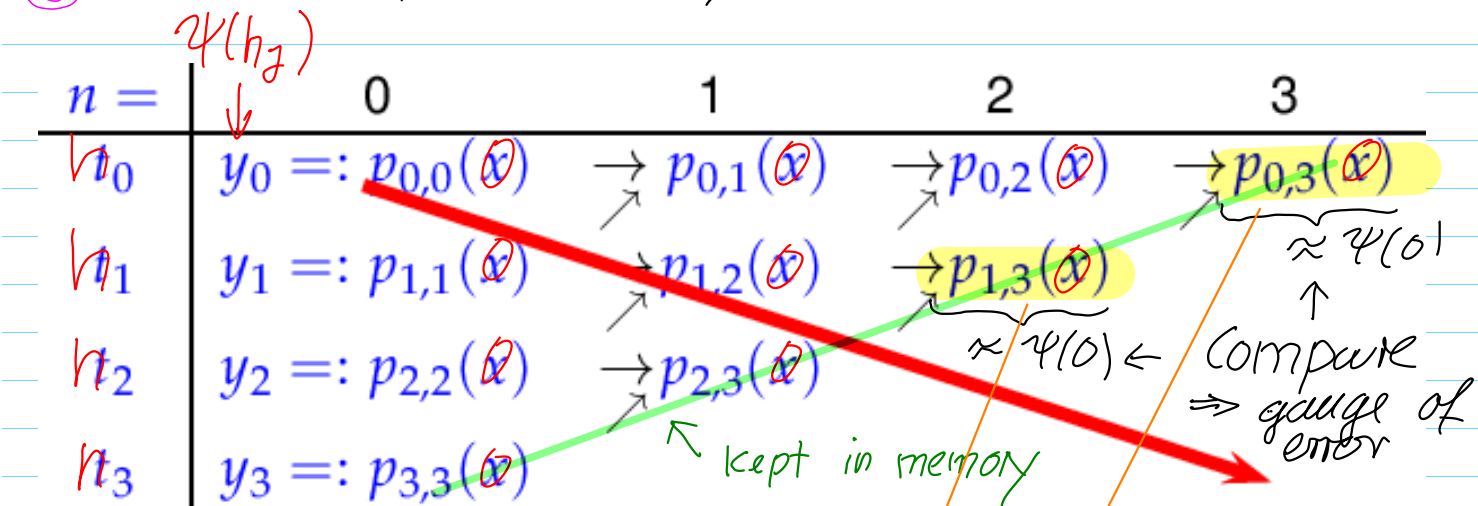
C++code 5.2.3.21: Polynomial evaluation

```

1 class PolyEval {
2     private:
3         // evaluation point and various internal data describing the
           polynomials
4     public:
5         // Idle Constructor
           PolyEval();
6         // Add another data point and update internal information
           void addPoint(t,y);
7         // Evaluation of current interpolating polynomial at x
           Eigen::VectorXd operator () (const Eigen::VectorXd &x) const;
8
9
10
11 };

```

→ add $(h_i, \psi(h_i))$

③ A.N. recursive ($n=3$)

C++-code 5.2.3.19: Numerical differentiation by adaptive extrapolation to zero

```

2 // Extrapolation based numerical differentiation
3 // with a posteriori error control
4 // f: handle of a function defined in a neighbourhood of  $x \in \mathbb{R}$ 
5 // x: point at which approximate derivative is desired
6 // h0: initial distance from x
7 // rtol: relative target tolerance, atol: absolute tolerance
8 template <class Function>
9 double diffex(Function& f, const double x, const double h0,
10               const double rtol, const double atol) {
11     const unsigned nit = 10; // Maximum depth of extrapolation
12     VectorXd h(nit); h[0] = h0; // Widths of difference quotients
13     VectorXd y(nit); // Approximations returned by difference quotients
14     y[0] = (f(x + h0) - f(x - h0))/(2*h0); // Widest difference quotients
15
16     // using Aitken-Neville scheme with  $x=0$ , see Code 5.2.3.10
17     for (unsigned i = 1; i < nit; ++i) {
18         // create data points for extrapolation
19         h[i] = h[i-1]/2; // Next width half a big
20         y[i] = (f(x + h[i]) - f(x - h[i]))/(2.0*h[i]); } Next data point ( $h_i, \psi(h_i)$ )
21         // Aitken-Neville update
22         for (int k = i - 1; k >= 0; --k)
23             y[k] = y[k+1] - (y[k+1]-y[k])*h[i]/(h[i]-h[k]); } next diagonal
24         // termination of extrapolation when desired tolerance is reached
25         const double errest = std::abs(y[1]-y[0]); // error indicator
26         if (errest < rtol*std::abs(y[0]) || errest < atol) // } stop, when estimated
27             break; } (relative) error  $\leq$  tolerance
28     }
29     return y[0]; // Return value extrapolated from largest number of
30 } difference quotients

```

```

auto f = [](double x)
{return std::atan(x);}
diffex(f, 1.1, 0.5)

```

```

auto g = [](double x)
{return std::sqrt(x);}
diffex(g, 1.1, 0.5)

```

```

auto h = [](double x)
{return std::exp(x);}
diffex(h, 1.1, 0.5)

```

Degree	Relative error
0	0.04262829970946
1	0.02044767428982
2	0.00051308519253
3	0.00004087236665
4	0.00000048930018
5	0.00000000746031
6	0.00000000001224

Degree	Relative error
0	0.02849215135713
1	0.01527790811946
2	0.00061205284652
3	0.00004936258481
4	0.00000067201034
5	0.00000001253250
6	0.00000000004816
7	0.00000000000021

Degree	Relative error
0	0.04219061098749
1	0.02129207652215
2	0.00011487434095
3	0.00000825582406
4	0.00000000589624
5	0.00000000009546
6	0.00000000000002

Enhanced accuracy + error control

▷ Review questions 5.2.3.20

4

5

7