COURSE 3

The Aitken's algorithm

Let $[a, b] \subset \mathbb{R}$, $x_i \in [a, b]$, $i = 0, 1, ..., m$ such that $x_i \neq x_j$ for $i \neq j$ and consider $f : [a, b] \to \mathbb{R}$.

Usually, for a practical approximation problem, for a given function $f : [a, b] \to \mathbb{R}$ we have to find the approximation of $f(\alpha)$, $\alpha \in [a, b]$ with an error not greater than a given $\varepsilon > 0$.

If we have enough information about f and its derivatives, we use the inequality $|(R_m f)(x)| \leq \varepsilon$ to find m such that $(L_m f)(\alpha)$ approximates $f(\alpha)$ with the given precision.

We may use the condition $\frac{|u(x)|}{(m+1)!} \left\| f^{(m+1)} \right\|_\infty \leq \varepsilon$, but it should be known $\left\| f^{(m+1)} \right\|_\infty$ or a majorant of it.

A practical method for computing the Lagrange polynomial is **the Aitken's algorithm.** This consists in generating the table:

$$
\begin{array}{c|c|c|ccccc}
x_0 & f_{00} & & & & & \\
x_1 & f_{10} & f_{11} & & & & \\
x_2 & f_{20} & f_{21} & f_{22} & & & \\
x_3 & f_{30} & f_{31} & f_{32} & f_{33} & & \\
\vdots & \vdots & \vdots & \vdots & \vdots & & \\
x_m & f_{m0} & f_{m1} & f_{m2} & f_{m3} & \cdots & f_{mm}
\end{array}
$$

where

$$f_{i0} = f(x_i), \quad i = 0, 1, ..., m,$$

and

$$f_{i,j+1} = \frac{1}{x_i - x_j} \begin{vmatrix} f_{jj} & x_j - x \\ f_{ij} & x_i - x \end{vmatrix}, \qquad i = 0, 1, ..., m; j = 0, ..., i - 1.$$

For example,

$$f_{11} = \frac{1}{x_1 - x_0} \begin{vmatrix} f_{00} & x_0 - x \\ f_{10} & x_1 - x \end{vmatrix}$$
$$= \frac{1}{x_1 - x_0} [f_{00}(x_1 - x) - f_{10}(x_0 - x)]$$
$$= \frac{x - x_1}{x_0 - x_1} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1) = (L_1 f)(x),$$

so f_{11} is the value in x of Lagrange polynomial for the nodes x_0, x_1. We have

$$f_{ii} = (L_i f)(x),$$

$L_i f$ being Lagrange polynomial for the nodes $x_0, x_1, ..., x_i$.

So $f_{11}, f_{22}, ..., f_{ii}, ..., f_{mm}$ is a sequence of approximations of $f(x)$.

If the interpolation procedure is convergent then the sequence is also convergent, i.e., $\lim\limits_{m \to \infty} f_{mm} = f(x)$. By Cauchy convergence criterion it follows

$$\lim_{i \to \infty} |f_{ii} - f_{i-1,i-1}| = 0.$$

This could be used as a stopping criterion, i.e.,

$$\left|f_{ii} - f_{i-1,i-1}\right| \leq \varepsilon, \quad \text{for a given precision } \varepsilon > 0.$$

Recommendation is to sort the nodes $x_0, x_1, ..., x_m$ with respect to the distance to x, such that

$$|x_i - x| \leq \left|x_j - x\right| \text{ if } i < j, \quad i, \quad j = 1, ..., m.$$

Example 1 *Approximate* $\sqrt{115}$ *with precision* $\varepsilon = 10^{-3}$, *using Aitken's algorithm.*

Newton interpolation polynomial

Let $[a, b] \subset \mathbb{R}$, $x_i \in [a, b]$, $i = 0, 1, ..., m$ such that $x_i \neq x_j$ for $i \neq j$ and consider $f : [a, b] \to \mathbb{R}$.

A useful representation for Lagrange interpolation polynomial is

$$(L_m f)(x) := (N_m f)(x) = f(x_0) + \sum_{i=1}^{m} (x - x_0)...(x - x_{i-1})(D^i f)(x_0) \tag{1}$$

$$= f(x_0) + \sum_{i=1}^{m} (x - x_0)...(x - x_{i-1})[x_0, ..., x_i; f],$$

which is called **Newton interpolation polynomial;** where $(D^i f)(x_0)$(or denoted $[x_0, ..., x_i; f]$) is the i-th order divided difference of the function f at x_0, given by the table

	f	$\mathcal{D}f$	$\mathcal{D}^2 f$	...	$\mathcal{D}^{\mathbf{m-1}} f$	$\mathcal{D}^m f$
x_0	f_0	$\mathcal{D}f_0$	$\mathcal{D}^2 f_0$	...	$\mathcal{D}^{m-1} f_0$	$\mathcal{D}^m f_0$
x_1	f_1	$\mathcal{D}f_1$	$\mathcal{D}^2 f_1$		$\mathcal{D}^{m-1} f_1$	
x_2	f_2	$\mathcal{D}f_2$	$\mathcal{D}^2 f_2$			
...	...	...				
x_{m-2}	f_{m-2}	$\mathcal{D}f_{m-2}$	$\mathcal{D}^2 f_{m-2}$			
x_{m-1}	f_{m-1}	$\mathcal{D}f_{m-1}$				
x_m	f_m					

Newton interpolation formula is

$$f = N_m f + R_m f,$$

where $R_m f$ denotes the remainder.

Assume that we add the point $(x, f(x))$ at the top of the table of divided differences:

	f	Df	$\dots$	$D^{m+1}f$
x	$f(x)$	$(Df)(x) = [x, x_0; f]$		$[x, x_0, \dots, x_m; f]$
x_0	$f(x_0)$	$(Df)(x_0) = [x_0, x_1; f]$	$\dots$	
x_1	$f(x_1)$	$(Df)(x_1) = [x_1, x_2; f]$		
$\dots$	$\dots$	$\dots$		
x_{m-1}	$f(x_{m-1})$	$(Df)(x_{m-1}) = [x_{m-1}, x_m; f]$		
x_m	$f(x_m)$			

For obtaining the interpolation polynomial we consider

$$[x, x_0; f] = \frac{f(x_0) - f(x)}{x_0 - x} \Longrightarrow f(x) = f(x_0) + (x - x_0)[x, x_0; f] \quad (2)$$

$$[x, x_0, x_1; f] = \frac{[x_0, x_1; f] - [x, x_0; f]}{x_1 - x} \quad (3)$$

$$\Longrightarrow [x, x_0; f] = [x_0, x_1; f] + (x - x_1)[x, x_0, x_1; f].$$

Inserting (3) in (2) we get

$$f(x) = f(x_0) + (x - x_0)[x_0, x_1; f] + (x - x_0)(x - x_1)[x, x_0, x_1; f].$$

If we continue eliminating the divided differences involving x in the same way, we get

$$f(x) = (N_m f)(x) + (R_m f)(x)$$

with

$$(N_m f)(x) = f(x_0) + \sum_{i=1}^{m} (x - x_0)...(x - x_{i-1})[x_0, ..., x_i; f]$$

and the remainder (the error) given by

$$(R_m f)(x) = (x - x_0)...(x - x_m)[x, x_0, ..., x_m; f]. \tag{4}$$

We notice that

$$(N_i f)(x) = (N_{i-1} f)(x) + (x - x_0)...(x - x_{i-1})[x_0, ..., x_i; f]$$

so the Newton polynomials of degree $2, 3, ...$, can be iteratively generated, similarly to Aitken's algorithm.

Example 2 *Find* $N_2 f$ *for* $f(x) = \sin \pi x$, *and the nodes* $x_0 = 0, x_1 = \frac{1}{6}, x_2 = \frac{1}{2}$.

Neville's algorithm

A problem of Lagrange interpolation is that the error term is quite difficult to be applied, so usually the degree of the suitable polynomial is not known until there have been made some computations.

Neville's method works with the approximating polynomials in a manner that uses previous calculations to greater advantage.

It is based on the Newton form of the interpolating polynomial and the recursion relation for the divided differences. It is similar to Aitken's algorithm.

Neville's algorithm for polynomial interpolation is widely used.

- The idea of the Neville's method is to use Lagrange polynomials of lower powers recursively in order to compute Lagrange polynomials of higher powers.

- Useful if you have the Lagrange polynomial based on some set of data points $(x_i, f(x_i)), i = 0, 1, \ldots, n$, and you get a new data point, $(x_{n+1}, f(x_{n+1}))$.

Definition 3 *Let f be a function defined at $x_0,\ x_1, \ldots,\ x_n$ and suppose that $m_1,\ \ldots,\ m_k$ are k distinct integers with $0 \leq m_i \leq n$, for every i. The Lagrange polynomial that interpolates $f(x)$ at the k points $x_{m_1}, \ldots,\ x_{m_k}$ is denoted by $P_{m_1,\ldots,m_k}(x)$.*

Example 4 *Consider $x_0 = 1$, $x_1 = 2$, $x_2 = 3$, $x_3 = 4$, $x_4 = 6$ and $f(x) = e^x$. Determine $P_{1,2,4}(x)$ and use it to approximate $f(5)$.*

$$\begin{aligned}P_{1,2,4}(x) &= \frac{(x-x_2)(x-x_4)}{(x_1-x_2)(x_1-x_4)}f(x_1) + \frac{(x-x_1)(x-x_4)}{(x_2-x_1)(x_2-x_4)}f(x_2) + \frac{(x-x_1)(x-x_2)}{(x_4-x_1)(x_4-x_1)}f(x_4)\\ &= \frac{(x-3)(x-6)}{(2-3)(2-6)}f(2) + \frac{(x-2)(x-6)}{(3-2)(3-6)}f(3) + \frac{(x-2)(x-3)}{(6-2)(6-3)}f(6)\end{aligned}$$

So, $f(5) \approx P_{1,2,4}(5)$, more specifically

$$\begin{aligned}P_{1,2,4}(5) &= \frac{(5-3)(5-6)}{(2-3)(2-6)}e^2 + \frac{(5-2)(5-6)}{(3-2)(3-6)}e^3 + \frac{(5-2)(5-3)}{(6-2)(6-3)}e^6\\ &= -0.5e^2 + e^3 + 0.5e^6 \approx 218.105.\end{aligned}$$

The following results present a method for recursively generating Lagrange's polynomial approximation.

Theorem 5 *Let f be a function defined at the points x_0, $x_1, \ldots,$ x_k and let x_i and x_j be two distinct points in this set. Then*

$$P(x) = \frac{(x - x_j)P_{0,1,\ldots,j-1,j+1,\ldots,k}(x) - (x - x_i)P_{0,1,\ldots,i-1,i+1,\ldots,k}(x)}{x_i - x_j}$$

is the kth Lagrange polynomial that approximates f at the $k+1$ nodes x_0, $x_1, \ldots,$ x_k.

Denote by $P_j(x) = f(x_j)$.

The previous Theorem implies that the interpolating polynomials can be generated recursively. For example

$$P_{0,1} = \frac{1}{x_1 - x_0}\left[(x - x_0)P_1 - (x - x_1)P_0\right]$$
$$P_{1,2} = \frac{1}{x_2 - x_1}\left[(x - x_1)P_2 - (x - x_2)P_1\right]$$
$$P_{0,1,2} = \frac{1}{x_2 - x_0}\left[(x - x_0)P_{1,2} - (x - x_2)P_{0,1}\right]$$

and so on. In the table below it can be seen how they are generated, taking into account the fact that each row is completed before the succeeding rows are begun.

x_0	P_0				
x_1	P_1	$P_{0,1}$			
x_2	P_2	$P_{1,2}$	$P_{0,1,2}$		
x_3	P_3	$P_{2,3}$	$P_{1,2,3}$	$P_{0,1,2,3}$	
x_4	P_4	$P_{3,4}$	$P_{2,3,4}$	$P_{1,2,3,4}$	$P_{0,1,2,3,4}$

This procedure is called the **Neville's method**.

- Going down in the table $\rightarrow$ using consecutive points x_i, with larger i

- Going to the right $\rightarrow$ increasing the degree of the interpolating polynomial.

- The points appear consecutively in each entry $\Longrightarrow$ we need to give only a starting point and the number of additional points used.

- Let $Q_{i,j}(x) = P_{i-j,i-j+1,\ldots,i-1,i}(x)$, with $0 \le j \le i$.

- The table becomes

x_0	$P_0 = Q_{0,0}$				
x_1	$P_1 = Q_{1,0}$	$P_{0,1} = Q_{1,1}$			
x_2	$P_2 = Q_{2,0}$	$P_{1,2} = Q_{2,1}$	$P_{0,1,2} = Q_{2,2}$		
x_3	$P_3 = Q_{3,0}$	$P_{2,3} = Q_{3,1}$	$P_{1,2,3} = Q_{3,2}$	$P_{0,1,2,3} = Q_{3,3}$	
x_4	$P_4 = Q_{4,0}$	$P_{3,4} = Q_{4,1}$	$P_{2,3,4} = Q_{4,2}$	$P_{1,2,3,4} = Q_{4,3}$	$P_{0,1,2,3,4} = Q_{4,4}$

with $Q_{i,j} = \frac{(x_i - x)Q_{i-1,j-1} + (x - x_{i-j})Q_{i,j-1}}{x_i - x_{i-j}}$, for $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, i$.

Neville's Iterated Interpolation Algorithm. Evaluate the polynomial P on $n+1$ given nodes, $x_0, \ldots, x_n$, at a given point x, for a function f.

Input : The nodes $x, x_0, x_1, \ldots, x_n$; the values of the function $f(x_0), \ldots, f(x_n)$ as the first column of Q $(Q_{0,0}, Q_{1,0}, \ldots, Q_{n,0})$.

Output : the table Q with $P(x) = Q_{n,n}$.

Step 1 for $i = 1, 2, \ldots, n$
for $j = 1, 2, \ldots, i$
$Q_{i,j} = \frac{(x_i - x)Q_{i-1,j-1} + (x - x_{i-j})Q_{i,j-1}}{x_i - x_{i-j}}$.

Step 2 Output(Q); Stop.

It can be additionally added a stopping criterion

$$|Q_{i,j} - Q_{i-1,j-1}| < \varepsilon,$$

with ε a given error tolerance. If the inequality is not fulfilled, a new interpolation node x_{i+1} is added.

Example 6 *Approximate* $\ln(2.1)$ *using the function* $f(x) = \ln x$ *and the data*

x	*2*	*2.2*	*2.3*
$\ln x$	*0.6931*	*0.7885*	*0.8329*

$\ln(2.1) = f(2.1)$ *and we have obtained the table*

```
Q =

   0.693100000000000                   0                   0
   0.788500000000000   0.740800000000000                   0
   0.832900000000000   0.744100000000000   0.741900000000000
```

So, $\ln(2.1) \approx 0.7419$, with an error of $3.7344 \cdot 10^{-5}$.

Example 7 *Approximate $\sqrt{115}$ using Neville's algorithm, with 3 given nodes.*

2.3. Hermite interpolation

Example 8 *In the following table there are some data regarding a moving car. We may estimate the position (and the speed) of the car when the time is $t = 10$ using Hermite interpolation.*

Time	0	3	5	8	13
Distance	0	225	383	623	993
Speed	75	77	80	74	72

Let $x_k \in [a, b],\ k = 0, 1, ..., m$ be such that $x_i \neq x_j$, for $i \neq j$ and let $r_k \in \mathbb{N},\ k = 0, 1, ..., m$. Consider $f : [a, b] \to \mathbb{R}$ such that there exist $f^{(j)}(x_k),\ k = 0, 1, ..., m;\ j = 0, 1, ..., r_k$ and $n = m + r_0 + ... + r_m$.

The Hermite interpolation problem (HIP) consists in determining the polynomial P of the smallest degree for which

$$P^{(j)}(x_k) = f^{(j)}(x_k), \quad k = 0, ..., m;\ j = 0, ..., r_k.$$

Definition 9 *A solution of (HIP) is called* **Hermite interpolation polynomial**, *denoted by* $H_n f$.

Hermite interpolation polynomial, $H_n f$, satisfies the interpolation conditions:

$$(H_n f)^{(j)}(x_k) = f^{(j)}(x_k), \quad k = 0, ..., m; \; j = 0, ..., r_k.$$

Hermite interpolation polynomial is given by

$$(H_n f)(x) = \sum_{k=0}^{m} \sum_{j=0}^{r_k} h_{kj}(x) f^{(j)}(x_k) \in \mathbb{P}_n, \tag{5}$$

where $h_{kj}(x)$ denote **the Hermite fundamental interpolation polynomials.** They fulfill the relations:

$h_{kj}^{(p)}(x_\nu) = 0, \; \nu \neq k, \quad p = 0, 1, ..., r_\nu$

$h_{kj}^{(p)}(x_k) = \delta_{jp}, \; p = 0, 1, ..., r_k, \quad$ for $j = 0, 1, ..., r_k$ and $\nu, k = 0, 1, ..., m$,

with $\delta_{jp} = \begin{cases} 1, & j = p \\ 0, & j \neq p. \end{cases}$

We denote by

$$u(x) = \prod_{k=0}^{m} (x - x_k)^{r_k+1} \qquad \text{and } u_k(x) = \frac{u(x)}{(x - x_k)^{r_k+1}}.$$

We have

$$h_{kj}(x) = \frac{(x - x_k)^j}{j!} u_k(x) \sum_{\upsilon=0}^{r_k-j} \frac{(x - x_k)^\nu}{\nu!} \left[\frac{1}{u_k(x)}\right]^{(\nu)}_{x=x_k}. \qquad (6)$$

Example 10 *Find the Hermite interpolation polynomial for a function f for which we know $f(0) = 1, f'(0) = 2$ and $f(1) = -3$ (equivalent with $x_0 = 0$ multiple node of order 2 or double node, $x_1 = 1$ simple node).*

Sol. We have $x_0 = 0, x_1 = 1, m = 1, r_0 = 1, r_1 = 0, n = m + r_0 + r_1 = 2$

$$
\begin{aligned}
(H_2 f)(x) &= \sum_{k=0}^{1} \sum_{j=0}^{r_k} h_{kj}(x) f^{(j)}(x_k) \\
&= h_{00}(x) f(0) + h_{01}(x) f'(0) + h_{10}(x) f(1).
\end{aligned}
$$

We have h_{00}, h_{01}, h_{10}. These fulfills relations:

$h_{kj}^{(p)}(x_\nu) = 0, \ \nu \neq k, \quad p = 0, 1, ..., r_\nu$

$h_{kj}^{(p)}(x_k) = \delta_{jp}, \ p = 0, 1, ..., r_k,$ for $j = 0, 1, ..., r_k$ and $\nu, k = 0, 1, ..., m.$

We have $h_{00}(x) = a_1 x^2 + b_1 x + c_1 \in \mathbb{P}_2$, with $a_1, b_1, c_1 \in \mathbb{R}$, and the system

$$
\begin{cases} h_{00}(x_0) = 1 \\ h'_{00}(x_0) = 0 \\ h_{00}(x_1) = 0 \end{cases} \Leftrightarrow \begin{cases} h_{00}(0) = 1 \\ h'_{00}(0) = 0 \\ h_{00}(1) = 0 \end{cases}
$$

that becomes

$$
\begin{cases} c_1 = 1 \\ b_1 = 0 \\ a_1 + b_1 + c_1 = 0. \end{cases}
$$

Solution is: $a_1 = -1, b_1 = 0, c_1 = 1$ so $h_{00}(x) = -x^2 + 1$.

We have $h_{01}(x) = a_2x^2 + b_2x + c_2 \in \mathbb{P}_2$, with $a_2, b_2, c_2 \in \mathbb{R}$. The system is

$$\begin{cases} h_{01}(x_0) = 0 \\ h'_{01}(x_0) = 1 \\ h_{01}(x_1) = 0 \end{cases} \Leftrightarrow \begin{cases} h_{01}(0) = 0 \\ h'_{01}(0) = 1 \\ h_{01}(1) = 0 \end{cases}$$

and we get $h_{01}(x) = -x^2 + x$.

We have $h_{10}(x) = a_3x^2 + b_3x + c_3 \in \mathbb{P}_2$, with $a_3, b_3, c_3 \in \mathbb{R}$. The system is

$$\begin{cases} h_{10}(x_0) = 0 \\ h'_{10}(x_0) = 0 \\ h_{10}(x_1) = 1 \end{cases} \Leftrightarrow \begin{cases} h_{10}(0) = 0 \\ h'_{10}(0) = 0 \\ h_{10}(1) = 1 \end{cases}$$

and we get $h_{10}(x) = x^2$.

The Hermite polynomial is

$$(H_2f)(x) = -x^2 + 1 - 2x^2 + 2x - 3x^2 = -6x^2 + 2x + 1.$$